OpenWIS 4 – Detailed Design for GeoNetwork Component

Contents

Work Package Description

Product Name: OpenWIS 4 – Detailed Design for GeoNetwork Component

Required by (completed, approved and signed-off): 4th September 2015

Approval method: design review

Background

The WMO Information System (WIS) provides an overarching approach to data and information management for all programmes within the World Meteorological Organisation (WMO) and related international organisations.

WIS leverages the long-standing collaborative culture of WMO. It is a globally distributed system within which WMO Members operate "WIS Centres": National Centres (NC), Data Collection and Production Centres (DCPC) and Global Information System Centres (GISC).

WIS comprises:

- a discovery metadata catalogue, synchronised between all GISCs;
- a cache of near real-time data products, synchronised between all GISCs;
- services facilitating subscription to, and delivery of, data products; and
- monitoring functions.

WIS metadata must comply with the WMO Core Metadata Profile- a profile of ISO 19115:2003.

More information on WIS and the WMO Core Metadata Profile can be found in WMO No 1060 "Manual on the WMO Information System" and WMO No 1061 "Guide to the WMO Information System".

The OpenWIS Association, a non-profit entity operating under Belgian Law of which the Met Office is a founding member, has developed software components (the "OpenWIS software") that implement the necessary functions required to operate a WIS Centre. The metadata management and catalogue functions of the current version of the OpenWIS software derive from GeoNetwork 2.6.

Given that GeoNetwork has undergone significant evolution since version 2.6, providing improved functionality for users / administrators, internal refactoring and incorporating many bug-fixes, a new version of the OpenWIS software (version 4) is planned that will (i) take advantage of the latest release of GeoNetwork, and (ii) be implemented such that subsequent releases of GeoNetwork are simple to (re-)integrate with the OpenWIS software.

Scope

Provide the detailed design for a set of enhancements to GeoNetwork that are required for the OpenWIS project. The design must provide sufficient detail for a competent software development team to implement without needing extensive knowledge of GeoNetwork's implementation.

These enhancements comprise two categories:

i. Those to be contributed to the main GeoNetwork project; and
ii. Those to be implemented as complementary components maintained within the OpenWIS code-base.

The enhancements described in this document often include new workflows; for example, to set up a new subscription for routine delivery of data. This document does not provide sufficient information to fully describe these workflows. The intent for this work package is to provide a *pattern* for the design that the software development team can embellish with the necessary level of detail during implementation.

Where enhancements are implemented as complementary components it is imperative that these are loosely coupled with GeoNetwork, thereby enabling these components to operate with new versions of GeoNetwork as these become available.

Given the extensive refactoring of GeoNetwork, version 3 is selected as the implementation target for these enhancements. If responders wish propose an alternative implementation target then rationale must be provided within the tender response.

Access to the OpenWIS source code can be provided on request should this be necessary.

Deliverables

The deliverables from this design work package include:

- Detailed design document
- Design review workshop (0.5-day)

The detailed design document:

- must provide sufficient detail for competent software development team to implement without needing extensive knowledge of GeoNetwork's implementation;
- should provide a *pattern* for the design where detailed information on workflows is absent; and
- should include implementation examples in order to clarify the design approach.

The design review workshop may be undertaken remotely (e.g. using video conferencing). A draft of the detailed design document must be provided at least one week prior to the design review workshop.

Overview of OpenWIS

The OpenWIS software comprises a number of sub-components- as illustrated in Figure 1.

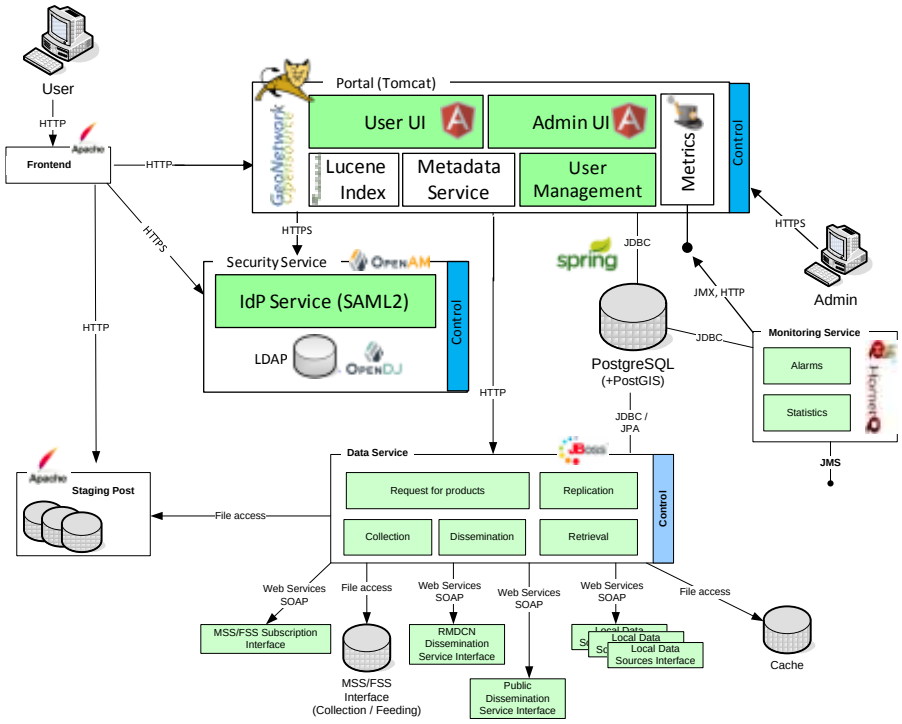

Figure 1 OpenWIS version 4 component architecture

The sub-components are:

- **Portal**: providing facilities for user authorization, data discovery, ad-hoc requests for data product delivery, metadata management, user management, subscription management and review of user's data product download histories.
- **Data Service**: managing the data products and fulfilling subscription services-
  - Collection and distribution of data products via the WMO Global Telecommunications System (GTS) via Message and/or File Switching Systems (MSS and FSS)
  - Maintaining the cache of near real-time data products arriving from the GTS
  - Maintaining the relationship between metadata records and specific instances of the data products they describe that may be in the cache or available in local data services – the 'Product Metadata Table'
  - Disseminating data products to users in accordance with ad-hoc requests and subscriptions
- **Security Service**: (optional) authentication service providing a SAML2 Identity Provider (IdP) Service.

- **Staging Post**: file-system where data products extracted from the cache or local data services are placed enabling them to be downloaded via HTTP.
- **Monitoring Service**: monitoring service quality and thresholds; issuing alerts / warnings when notifiable events occur (e.g. when a user exceeds their data download thresholds and is 'blacklisted').

The WIS catalogue will include metadata records that describe data products that are available neither from the cache nor local data services[1]. In these cases, the portal must not offer the facility to download the data product or set up a subscription for routine delivery.

Communication between components uses SOAP web services. As OpenWIS evolves, the inter-component communication will likely move away from this technology choice. However, for the purposes of this work package, the existing SOAP web services shall be used. [Annex A: WSDL descriptions of OpenWIS web services] lists the operations provided by each SOAP web service. Full WSDL descriptions and associated resources are also provided in the zip file 'OpenWIS WSDL and XSD' as well as the OpenWIS github repo: /openwis-metadataportal/openwis-portal-client/src/wsdl/dataservice/.

Metadata management

Unless explicitly stated below, the Met Office will contribute the following functionality to the GeoNetwork project. In these cases, source code required to implement these functions must be implemented such that it could be incorporated into the main GeoNetwork trunk.

Enhancement MM1: Use of custom metadata file identifier

GeoNetwork uses a UUID for the metadata file identifier for a metadata record. The WMO Core Metadata Profile requires that metadata records are identified using a URN that follows a specific pattern.

GeoNetwork shall be amended such that:

i. The metadata file identifier for a metadata record can be specified as a URN
ii. An exception shall be thrown if the metadata file identifier is not unique; the uniqueness of metadata file identifier URNs is *case insensitive*
iii. A pattern for the metadata file identifier URN can be configured based on RFC 6570 URI Template

Enhancement MM2: Configurable limit on OAI-PMH metadata harvest response

GeoNetwork shall be amended such that the number of metadata records returned in response to an OAI-PMH harvest request is limited to a configurable number.

Enhancement MM3: Metadata categories and category-based harvesting

In order to provide greater control in the harvesting of metadata between WMO Information System Centres (WIS Centres), OpenWIS provides the facility to group metadata records into named subsets, or categories. Each category of metadata records can be offered for harvesting by a third party as a distinct subset of the total set of metadata records within the OpenWIS catalogue.

[1] Data products described in the WIS catalogue may be available from many channels, including out-of-band delivery, web services that are external to the OpenWIS system etc.

GeoNetwork shall be amended such that:

i. A metadata record may be assigned to a category
ii. Metadata harvesting can be configured for each metadata category; e.g. specific metadata harvesting configurations can be specified for the categories of metadata, enabling OpenWIS to expose named subsets of metadata for harvesting by third parties.
iii. The GeoNetwork role "Administrator" can define new categories.
iv. The GeoNetwork role "Administrator" can modify category assignment of *published* metadata records[2]
v. The GeoNetwork role "User Administrator" can assign the permission to use categories to specific Groups and define a default category for each Group.
vi. The GeoNetwork role "Editor" can assign a metadata record that they have permission to modify to a specific category.

Note that permissions are assumed to cascade in the order: Guest (non-authenticated user), Registered User, Editor, Content Reviewer, User Administrator and Administrator. For example, a user with role Content Reviewer can perform all the actions assigned to a user with role Editor plus a few more.

Enhancement MM4: Maintain product metadata table

This enhancement is to be implemented as a complementary component to GeoNetwork as the functionality described herein is specific to OpenWIS. Implementation of this enhancement should be loosely coupled with the GeoNetwork code base enabling the source code for this enhancement to be maintained independently. It is highly desirable to be able to bind the capabilities described by this enhancement within GeoNetwork using deployment-time configuration rather than at compilation-time.

The OpenWIS Data Service holds a summary record for each metadata record in the catalogue. This information is stored in the Product Metadata Table and drives the processing of the Data Service on arrival and dissemination of data product instances.

The definition of the summary record, expressed as a *complex type* within XML Schema, is provided below:

```
<xs:complexType name="productMetadata">
  <xs:sequence>
    <xs:element minOccurs="0" name="id" type="xs:long"/>
    <xs:element minOccurs="0" name="creationDate" type="xs:dateTime"/>
    <xs:element minOccurs="0" name="urn" type="xs:string"/>
    <xs:element minOccurs="0" name="dataPolicy" type="xs:string"/>
    <xs:element minOccurs="0" name="fed" type="xs:boolean"/>
    <xs:element minOccurs="0" name="fncPattern" type="xs:string"/>
    <xs:element minOccurs="0" name="gtsCategory" type="xs:string"/>
    <xs:element minOccurs="0" name="ingested" type="xs:boolean"/>
    <xs:element minOccurs="0" name="localDataSource" type="xs:string"/>
    <xs:element minOccurs="0" name="originator" type="xs:string"/>
    <xs:element minOccurs="0" name="overridenDataPolicy" type="xs:string"/>
    <xs:element minOccurs="0" name="overridenGtsCategory" type="xs:string"/>
    <xs:element minOccurs="0" name="overridenFncPattern" type="xs:string"/>
    <xs:element minOccurs="0" name="overridenPriority" type="xs:int"/>
```

[2] Modification of category assignment for published metadata records is restricted to Administrators as this has an impact on the harvesting.

```
      <xs:element minOccurs="0" name="priority" type="xs:int"/>
      <xs:element minOccurs="0" name="process" type="xs:string"/>
      <xs:element minOccurs="0" name="title" type="xs:string"/>
      <xs:element minOccurs="0" name="fileExtension" type="xs:string"/>
      <xs:element minOccurs="0" name="overridenFileExtension" type="xs:string"/>
      <xs:element minOccurs="0" name="updateFrequency" type="tns:updateFrequency"/>
      <xs:element minOccurs="0" name="stopGap" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
```

Information for the summary record is located within the main metadata record according to XPath statements.

When a new metadata record is added to the OpenWIS catalogue, a corresponding entry in the Product Metadata Table shall be added using the `createProductMetadata` operation of the `ProductMetadataService` SOAP web service.

When an existing metadata record is updated within the OpenWIS catalogue, the corresponding entry in the Product Metadata Table shall be updated using the `updateProductMetadata` operation of the `ProductMetadataService` SOAP web service.

When an existing metadata record is deleted within the OpenWIS catalogue, the corresponding entry in the Product Metadata Table shall be removed using the `deleteProductMetadata`, `deleteProductMetdataByURN` or `deleteProductMetadatasWithURN` operations of the `ProductMetadataService` SOAP web service. The latter operation provides for bulk removal.

Note that the obsoleting of subscriptions associated with a deleted metadata record is undertaken by the OpenWIS Data Service.

Authentication and access control

The Met Office will contribute the following functionality to the GeoNetwork project. Source code required to implement these functions must be implemented such that it could be incorporated into the main GeoNetwork trunk.

Previous versions of the OpenWIS software have implemented a policy-based access control; wherein data policies are assigned to metadata records and their associated data resources. However, as part of the move to OpenWIS version 4, the existing authorization implemented within GeoNetwork will be used to provide role-based access control[3]. As such, the permissions *publish*, *download*, *interactive map*, *featured*, *edit* and *notify* will be assigned to specified Groups for a given metadata record- noting that the assignment of permissions may be applied to a set of metadata records in a single action in order to simplify management workflows. Users with role *Editor* or *Content Reviewer* for a given metadata record are ultimately responsible for determining which permissions get assigned to that metadata record.

[3] At some future point, the OpenWIS Association may investigate how metadata records expressing well-known data policies such as “WMO Essential” and “WMO Additional”, as defined at XPath `/gmd:MD_Metadata/gmd:identificationInfo//gmd:resourceConstraints/gmd:MD_LegalConstaints/gmd:otherConstraints//`, may automatically be associated with a specified set of permissions.

Enhancement AAC1: SAML2 authentication

GeoNetwork shall be amended such that user authentication may be provided by a SAML2 Identity Provider (IdP) service. Use of SAML2 for authentication must be configurable according to the needs of a given deployment.

A given GeoNetwork deployment may authenticate via multiple SAML2 IdP service end-points- noting that a specific Identity for an individual user will be associated with only one SAML2 IdP.

There is no requirement to retrieve any attributes about the identified user from the SAML2 IdP.

Note that the Spring Security project used within GeoNetwork provides a SAML extension.

Enhancement AAC2: Assign SAML Identity to existing user

GeoNetwork shall be amended such that, where a deployment is configured to make use of SAML2 for authentication, an existing user may be associated with a specified SAML Identity from one of the configured SAML2 IdP services.

Management of the user's profile will continue to be undertaken within GeoNetwork- only the authentication method is amended.

There is no requirement to retain the original authentication mechanism (e.g. username and password) for that user.

Note that where LDAP is used for user authentication *and* the LDAP provides user attributes, it may not be possible to configure users for SAML authentication.

Enhancement AAC3: New user with SAML Identity

GeoNetwork shall be amended such that, where a deployment is configured to make use of SAML2 for authentication, a new user may be created and associated with a specified SAML Identity from one of the configured SAML2 IdP services.

Management of the user's profile will be undertaken within GeoNetwork- only authentication is provided by the SAML2 IdP.

Data download, ad-hoc delivery and subscription

These enhancements are to be implemented as one or more complementary components to GeoNetwork as the functionality described herein is specific to OpenWIS. Implementation of this enhancement should be loosely coupled with the GeoNetwork code base enabling the source code for this enhancement to be maintained independently. It is highly desirable to be able to bind the capabilities described by this enhancement within GeoNetwork using deployment-time configuration rather than at compilation-time.

Following a search within the catalogue, a user may choose to download the data described by the selected metadata record if they have the necessary *download* permissions and they are not *blacklisted*[4].

If the dataset is available from the OpenWIS Data Service, zero or more instances of the associated data product may be available for download. The user is able to select those data product instances that they require and request their extraction from the cache or local data service.

Download may occur directly via the portal (HTTP download) or as an ad-hoc request to deliver via email, FTP or some other dissemination mechanism offered by the OpenWIS Data Service.

Furthermore, a user may set up a subscription to that dataset providing routine delivery of data product instances via email, FTP etc. as they become available to OpenWIS.

Note that internally, the OpenWIS Data Service uses message queues operating in two stages. Data product download and new subscription requests are queued first for validation. A second queue accepts *processed requests*- either validated download request or requests arising from the OpenWIS Data Service processing an existing subscription. Data product instances are extracted from the cache or local data service, and packaged as necessary, on processing of a message from the Processed Request queue. Once extracted and packaged, the data products are moved to the Staging Post from where either the user can download the data product package directly via HTTP or the OpenWIS Data Service disseminates the file via the user's selected mechanism (e.g. email, FTP etc.) to the designated address. Importantly, the HTTP URL of the data product package is not known until the extraction process is complete and the resulting file is moved to the Staging Post.

Enhancement DDS1: Download offer

GeoNetwork shall be amended such that, when the conditions outlined below are met, a user is presented with the opportunity to download or subscribe to the dataset described by the metadata record they are viewing.

The user may subscribe to the dataset under the following conditions:

1. The user has the necessary *download* permissions for the dataset
2. The dataset is managed by the OpenWIS Data Service (either cache or a local data service)- as determined via attributes from the summary record within the Product Metadata Table[5].

A "subscribe" option shall be provided by the portal enabling a new subscription to be requested (see enhancement *DDS2* below).

Furthermore, the user may download or request immediate delivery of the dataset if the following conditions are *also* met:

[4] A user may be blacklisted if they have exceed their data download thresholds or if they are deemed to have breached the fair usage policy by a system administrator.

[5] The summary record from the Product Metadata Table can be obtained using the operations such as `getProductMetadataByUrn` (where the URN is the metadata file identifier) from the `ProductMetadataService` SOAP web-service.

3. The user is *not* blacklisted [6](e.g. they have not already exceeded their daily download threshold)
4. Data product instances are currently available[7].

Options "deliver" (see enhancement *DDS3* below) and "download" (see enhancement *DDS4* below) shall be provided by the portal.

Enhancement DDS2: New subscription

GeoNetwork shall be amended such that a user with the appropriate permissions may set up a new subscription requesting routine delivery of data products from the OpenWIS Data Service.

The portal will provide a web-based user interface enabling a user to set up their subscription[8]; allowing them to specify details of their primary and secondary dissemination channels and how individual data product instances are packaged for delivery. On completion of the subscription details, the portal will package the user supplied information and the relevant product metadata record (retrieved via the `ProductMetadataService` SOAP web-service) into a subscription request object and send it for validation and processing to the OpenWIS Data Service using the `createSubscription` operation from the `SubscriptionService` SOAP web-service.

OpenWIS shall notify the relevant Groups that a new subscription has been created according to the *notify* permission defined for the metadata record associated with the dataset that has been subscribed to.

Enhancement DDS3: Ad-hoc request for delivery

GeoNetwork shall be amended such that a user with the appropriate permissions may request the immediate delivery of data product instances that are currently available from the OpenWIS Data Service; so-called "ad-hoc" delivery.

The portal will provide a web-based user interface enabling a user to configure their ad-hoc request for the specified dataset; allowing them to specify details of their chosen dissemination channel, how individual data product instances are packaged for delivery and which specific data product instances are required [9]. On completion of the ad-hoc request details, the portal will package the user supplied information and the relevant product metadata record (retrieved via the `ProductMetadataService` SOAP web-service) into an ad-hoc request object and send it for validation and processing to the OpenWIS Data Service using the `createRequest` operation from the `RequestService` SOAP web-service.

---

[6] Determining whether a user is blacklisted is achieved using the `isUserBlacklisted` operation of the `BlacklistService` SOAP web-service.

[7] Determining whether data product instances are currently available is achieved via the use of the SOAP web-services. For example, if data products are stored in the cache, the operation `listFilesByMetadataUrnAndDate` from the `CacheIndexService` SOAP web-service will provide a list of those instances currently in the cache associated with the designated metadata record.

[8] The current OpenWIS software enables subscription requests to be parameterised. For OpenWIS version 4, the parameterisation choices offered will be driven according to the information in the metadata record. For the purposes of this technical design, we will ignore parameterisation of subscriptions, but note that this requirement will need to be incorporated at a later stage in the OpenWIS 4 development.

[9] The list of available data product instances can be retrieved using the `listFilesByMetadataUrnAndDate` operation from the `CacheIndexService` SOAP web-service.

OpenWIS shall notify the relevant Groups that the dataset has been downloaded according to the *notify* permission defined for the metadata record associated with the dataset that has been requested.

Enhancement DDS4: Direct download via HTTP

GeoNetwork shall be amended such that a user with the appropriate permissions may download data product instances that are currently available from the OpenWIS Data Service directly from the portal via HTTP.

The portal will provide a web-based user interface enabling a user to configure their download request for the specified dataset; allowing them to select specific data product instances from the currently available set[9] and specify how these are to be packaged for download (e.g. as a single compressed file, or concatenated according to a WMO–specified scheme).

Direct downloads are then processed identically to ad-hoc requests; the portal will package the user supplied information and the relevant product metadata record (retrieved via the `ProductMetadataService` SOAP web-service) into an ad-hoc request object and send it for validation and processing to the OpenWIS Data Service using the `createRequest` operation from the `RequestService` SOAP web-service.

Note that by re-using the existing services for ad-hoc requests, all the necessary usage statistics and download history are automatically gathered for that user. The absence of dissemination information within the request object means that no further action will be taken within the OpenWIS Data Service once the data products have been extracted, packaged and moved to the Staging Post.

Comment [J1]: It may be necessary to create a new sub-type of request for this action- but for this technical design work package, it is sufficient to assume that the absence of dissemination information will prevent any dissemination or email notification occurring.

The asynchronous nature of the ad-hoc request process means that it is not straightforward to determine when the packaged data products have been moved to the Staging Post ready for download. For the purposes of this technical design the `monitorExtraction` operation of the `ProcessedRequestService` SOAP web service can be used to poll for completion[10].

Once the ad-hoc request process has completed, the portal will be updated to provide a hyperlink to the location of the packaged data products within the Staging Post. The URL is determined by concatenating the Staging Post URL (retrieved from configuration) and the URI for the data product package (retrieved using the `getUri()` operation on the processed request object returned from the `getProcessedRequest` or `getProcessedRequestForAdhoc` operations from the `ProcessedRequestService` SOAP web-service).The user is then able to download the packaged data products.

OpenWIS shall notify the relevant Groups that the dataset has been downloaded according to the *notify* permission defined for the metadata record associated with the downloaded dataset.

Additional user interface requirements

These enhancements are to be implemented as one or more complementary components to GeoNetwork as the functionality described herein is specific to OpenWIS. Implementation of this

[10] A more elegant solution to identify when the ad-hoc request process has completed (involving call-backs, for example) will be implemented in due course.

enhancement should be loosely coupled with the GeoNetwork code base enabling the source code for this enhancement to be maintained independently. It is highly desirable to be able to bind the capabilities described by this enhancement within GeoNetwork using deployment-time configuration rather than at compilation-time.

Users must be able to manage their subscriptions and monitor the status of their downloads- direct via HTTP, ad-hoc delivery and routine delivery.

Given the absence of details for this section, the intent is for the technical design document to provide a reusable pattern that can be implemented to meet the enhancements listed below.

Enhancement UI1: Subscription management

GeoNetwork shall be amended to provide users with a web-based mechanism to review and manage (create, update and delete) their subscriptions for routine delivery of data.

The `SubscriptionService` SOAP web-service provides for retrieval and persistence of the subscription objects.

Enhancement UI2: "My-downloads" page

GeoNetwork shall be amended to provide users with a web-based mechanism to:

i. monitor the status of their active downloads;
ii. discard requests that are no longer required;
iii. browse their download history; and
iv. retrieve packaged data products from the Staging Post if they are still available.

The `RequestService` and `ProcessedRequestService` SOAP web-services provide for retrieval and persistence of the request and process-request objects.

Details of active downloads should automatically be refreshed.

Additional administration interface requirements

These enhancements are to be implemented as one or more complementary components to GeoNetwork as the functionality described herein is specific to OpenWIS. Implementation of this enhancement should be loosely coupled with the GeoNetwork code base enabling the source code for this enhancement to be maintained independently. It is highly desirable to be able to bind the capabilities described by this enhancement within GeoNetwork using deployment-time configuration rather than at compilation-time.

OpenWIS administrators require additional tools in order to manage the OpenWIS system effectively. These are listed in the enhancements below.

Given the absence of details for this section, the intent is for the technical design document to provide a reusable pattern that can be implemented to meet the enhancements listed below. It is anticipated that these functions would be implemented as new panels within the GeoNetwork Administration Console.

Enhancement ADMIN1: Blacklisting

A black-listed user is able to browse the catalogue but is not able to download data from OpenWIS nor are their subscriptions fulfilled. Black-listing occurs when a user's data download thresholds have been exceeded within a given period (e.g. 24-hours), or when an administrator deems that a given user has breached the fair-usage policy. A second threshold is defined for each user that is used to trigger delivery of warning to that user indicating their imminent breach of the data download threshold.

GeoNetwork shall be amended to provide a User Administrator with a web-based mechanism to:

i. blacklist a specific user;
ii. blacklist all users within a specific Group;
iii. revoke blacklisting for a specific user;
iv. revoke blacklisting for all users within a specific Group;
v. amend the download thresholds for a specific user or for all users within a specific Group; and
vi. review blacklist thresholds and download volumes for the current period for all users or for all users within a specific Group.

The `BlacklistService` SOAP web-service provides for retrieval and persistence of black-list information.

Enhancement ADMIN2: Browse product metadata table

Geonetwork shall be amended to provide an Administrator with a web based mechanism to:

i. browse the content of the product metadata table; and
ii. for specific records, override attributes (e.g. dissemination priority) that have been extracted from the associated metadata record in the catalogue.

The ProductMetadataService SOAP web-service provides a mechanism to interact with the product metadata table.

Enhancement ADMIN3: Browse OpenWIS cache content

GeoNetwork shall be amended to provide an Administrator with a web-based mechanism to:

i. browse the content of the global cache; and
ii. extract specific data products from the cache.

The `CacheIndexService` SOAP web-service provides a mechanism to interact with the cache.

Enhancement ADMIN4: Monitoring and statistics

GeoNetwork shall be amended to display the system information and statistics available via the `MonitoringService` SOAP web-service.

System information and statistics should be exportable in a machine readable form.

Enhancement ADMIN5: Configure dissemination channels for Groups

An OpenWIS deployment will provide a number of channels for disseminating data products to users via subscription or ad-hoc delivery request (e.g. email, FTP etc.).

The available dissemination channels shall be specified in as configuration items.

GeoNetwork shall be amended such that a User Administrator can assign to a Group the permission to use a dissemination channel. A Group may be permitted to use multiple dissemination channels. At a minimum, all Groups are permitted to disseminate data products via Staging Post- in which case an email notification is generated indicating the URL of the data product that has been generated.

Enhancement ADMIN6: Browse subscriptions

GeoNetwork shall be amended to provide a User Administrator with a web-based mechanism to:

i. browse the subscriptions for all users or users within a specific Group; and
ii. suspend and reactivate specific subscriptions.

The `SubscriptionService` SOAP web-service provides for retrieval and persistence of the subscription objects.

Default metadata

The Met Office will contribute the following functionality to the GeoNetwork project should it not already exist. Source code required to implement these functions must be implemented such that it could be incorporated into the main GeoNetwork trunk.

OpenWIS may receive data product instances that have no associated metadata record in the catalogue. In such a situation, the OpenWIS Data Service will create a default metadata record that describes the data product and insert this into the catalogue.

Default metadata records must be visible only to administrators (e.g. the records remain in an *unpublished* status).

Enhancement DM1: Default metadata insertion

GeoNetwork shall be amended such that a software agent is able to insert a metadata record into the catalogue wherein it is associated with a specific Group and its status remains *unpublished*.

Annex A: WSDL descriptions of OpenWIS web services

Blacklist service

BlacklistService.wsdl

- wsdl:operation name="getUsersBlackListInfo"
- wsdl:operation name="updateUserBlackListInfo"
- wsdl:operation name="getUserBlackListInfo"
- wsdl:operation name="isUserBlacklisted"
- wsdl:operation name="getUsersBlackListInfoByUser"
- wsdl:operation name="setUserBlacklisted"
- wsdl:operation name="getUserBlackListInfoIfExists"
- wsdl:operation name="checkAndUpdateDisseminatedData"

Cache index service

CacheIndexService.wsdl

- wsdl:operation name="getAllMetadataUrnsForCachedFile"
- wsdl:operation name="listAllCachedFiles"
- wsdl:operation name="listFilesByMetadataUrnAndTime"
- wsdl:operation name="getCacheContentCount"
- wsdl:operation name="getCachedFile"
- wsdl:operation name="addCacheIndexEntry"
- wsdl:operation name="listCachedFilesBetweenDates"
- wsdl:operation name="listCachedFiles"
- wsdl:operation name="getCacheContentSorted"
- wsdl:operation name="listFilesByMetadataUrnAndDate"
- wsdl:operation name="getCacheContent"
- wsdl:operation name="ping"
- wsdl:operation name="getBackupLastCollectDate"
- wsdl:operation name="setLastCollectDate"
- wsdl:operation name="backupLastCollectDate"
- wsdl:operation name="getCacheContentFilteredSorted"
- wsdl:operation name="getCachedFileById"

Monitoring service

MonitoringService.wsdl

- wsdl:operation name="GetRecentEvents"
  - The recent events (alarms) gathered by the system
- wsdl:operation name="GetGlobalReports"
  - The statistics (reports) gathered by the system
- wsdl:operation name="GetExchangedDataStatistics"
  - Volume of data disseminated and extracted per day
- wsdl:operation name="GetDisseminatedDataStatistics”
  - Volume of data disseminated per day and per user
- wsdl:operation name="GetIngestedDataStatistics”
  - Volume of data ingested per day
- wsdl:operation name="GetCatalogStatistics"
- wsdl:operation name="GetCacheStatistics"
- wsdl:operation name="GetCacheContents"

Processed request service

ProcessedRequestService.wsdl

- wsdl:operation name="getFullProcessedRequest"
- wsdl:operation name="getAllProcessedRequestsByUsers"
- wsdl:operation name="extract"
- wsdl:operation name="addProcessedRequestToSubscription"
- wsdl:operation name="getProcessedRequestForAdhoc"
- wsdl:operation name="getProcessedRequest"
- wsdl:operation name="getAllProcessedRequestsByRequestCount"
- wsdl:operation name="getAllProcessedRequestsByUsersCount"
- wsdl:operation name="clearProcessedRequestStagingPost"
- wsdl:operation name="getFullProcessedRequestForAdhoc"
- wsdl:operation name="deleteProcessedRequestsByRequest"
- wsdl:operation name="deleteProcessedRequests"
- wsdl:operation name="clearProcessedRequestStagingPostByUri"
- wsdl:operation name="deleteProcessedRequestWithAdHoc"
- wsdl:operation name="updateProcessedRequest"
- wsdl:operation name="monitorExtraction"
- wsdl:operation name="getAllProcessedRequestsByRequest"

Product metadata service

ProductMetadataService.wsdl

- wsdl:operation name="getProductMetadataById"
- wsdl:operation name="deleteProductMetadataByURN"
- wsdl:operation name="getLastStopGapMetadata"
- wsdl:operation name="getAllPatternMetadataMapping"
- wsdl:operation name="deleteProductMetadata"
- wsdl:operation name="getProductsMetadataByUrns"
- wsdl:operation name="deleteProductMetadatasWithURN"
- wsdl:operation name="createStopGapMetadata"
- wsdl:operation name="updateProductMetadata"
- wsdl:operation name="createProductMetadata"
- wsdl:operation name="getProductsMetadataCount"
- wsdl:operation name="getAllProductsMetadata"
- wsdl:operation name="getProductMetadataByUrn"

Request service

RequestService.wsdl

- wsdl:operation name="getRequestsByUsersCount"
- wsdl:operation name="getLastProcessedRequest"
- wsdl:operation name="getRequestsByUsers"
- wsdl:operation name="createRequest"
- wsdl:operation name="deleteRequest"
- wsdl:operation name="getRequest"
- wsdl:operation name="deleteRequestByUser"

Subscription service

SubscriptionService.wsdl

- wsdl:operation name="createSubscription"
- wsdl:operation name="setBackup"

- wsdl:operation name="checkUserSubscription"
- wsdl:operation name="getSubscription"
- wsdl:operation name="getSubscriptionsByUsersCount"
- wsdl:operation name="findLastProcessedRequest"
- wsdl:operation name="resumeSubscription"
- wsdl:operation name="getSubscriptionsByUsers"
- wsdl:operation name="getFullSubscription"
- wsdl:operation name="updateSubscription"
- wsdl:operation name="suspendSubscription"
- wsdl:operation name="deleteSubscription"
- wsdl:operation name="checkUsersSubscription"
- wsdl:operation name="updateSubscriptionConfig"