CR-Form-v3

CHANGE REQUEST

⌘ **WCS CR 098** ⌘ rev **-** ⌘ Current version: 1.0 ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ AS [ ] Imp. Stand [X] Best Practices Paper [ ] Other [ ]

Title: ⌘ Proposal for WCS Transactional - WCS-T

Source: ⌘ **Michael P. Gerlek (editor), LizardTech**

Name & email: ⌘ mpg@lizardtech.com ***Date:*** ⌘ 11/16/06

Category: ⌘ **B**

Use one of the following categories:
- ***F*** *(Critical correction)*
- ***A*** *(corresponds to a correction in an earlier release)*
- ***B*** *(Addition of feature),*
- ***C*** *(Functional modification of feature)*
- ***D*** *(Editorial modification)*

Reason for change: ⌘ Version 1.0 of the Web Coverage Service (WCS) Specification does not address how coverage data gets added to or deleted from a server

Summary of change: ⌘ See attached document

Consequences if not approved: ⌘ No transactional capability

Clause(s) affected: ⌘ See document below

Other specs Affected: ⌘ [ ] Other core specifications ⌘ [ ] Abstract specifications [ ] Best Practices Document

Supporting Doc. ⌘ Attached

Other comments: ⌘

Status ⌘

Disposition ⌘

-

1. Introduction

Version 1.0 of the Web Coverage Service (WCS) Specification does not address how coverage data gets added to or deleted from a server; it is assumed that some implementation-specific process exists for handling this, likely on the back end (server-side). One of the goals of OWS-4 was to extend WCS to support these operations, generally referred to as "WCS-T".

Early on in the OWS-4 process, however, the WSC RWG independently produced a Change Request targeted against the WCS 1.1 effort for the addition of a "harvest" operation. This change (document 06-043, by Arliss Whiteside) initially described how a client could pass a server a coverage description and a URL pointing to some coverage data. It did not originally consider the other operations or address adding coverage data directly, performing update operations, or deleting data.

As a result of the OWS-4 project, the 06-043 document has evolved to incorporate the larger WCS-T goals under WCS version 1.1. (Earlier drafts of this document discussed '043 and the changes required.)

Due to time constraints, LizardTech's implementation is not able to support the full 06-043 specification. Instead, we have chosen to implement a subset of the 06-043 functionality under WCS 1.0, as a proof of concept. This document gives the specification for this WCS 1.0 implementation; it also provides some discussion of certain general WCS-T issues.

1.1 References

The following OGC documents directly support or are indirectly related to this report:

- 05-076 – WCS 1.0 specification
- 06-043r3 – WCS Change Request: *"add Transaction operation"* (Arliss Whiteside)

1.2 Contributors

The following organizations and individuals contributed to this report:

LizardTech: Matt Fleagle, Sean Forde, Michael P. Gerlek, Chris Haukap, John Hayes, Glen Thompson

PCI Geomatics: Steven Keens

BAE Systems: Arliss Whiteside

2. WCS-T Changes for WCS 1.0

This section reflects the actual changes to the WCS 1.0 specification in order to support a simplified version of the 06-043 model.

2.1 Summary of Changes

Changes to `GetCapabilities`:

- `AddCoverage` (addition)
- `UpdateCoverageMetadata` (addition)
- `UpdateCoverage` (addition)
- `DeleteCoverage` (addition)
- `SupportedInputFormats` (addition)
- `SupportedInputCRSs` (addition)

Changes to `DescribeCoverage`:

- *(none)*

Changes to `GetCoverage`:

- *(none)*

Addition of `AddCoverage`.

Addition of `UpdateCoverageMetadata`.

Addition of `ReplaceCoverage`.

Addition of `DeleteCoverage`.

2.2 `GetCapabilities` Changes

2.2.1 Addition of New Request Operations

Four new capabilities request types are added **(change to WCS 1.0, section 7.3.3.7)**:

- `AddCoverage`
- `UpdateCoverageMetadata`
- `UpdateCoverage`
- `DeleteCoverage`

Note that these four operations are independent of each other; for example, a server may support coverage replacement and metadata updates, but not support coverage addition or deletion.

2.2.2 Addition of `SupportedInputFormats`

The WCS-T server shall specify the input formats it supports for the `AddCoverage` transaction request. In the Capabilities document, `SupportedInputFormats` will be located in `WCS_Capabilities / Capability`, parallel to the Request element. It will contain any number of `inputFormat` elements, one for each format the server supports.

Implementation Note: The LizardTech WCS-T server does not provide a list of supported input formats. It supports the following formats:

```
image/jp2;subtype="gmljp2"
image/jp2
```

2.2.3 Addition of `SupportedInputCRSs`

The WCS-T server shall specify the input CRSs it supports for the `AddCoverage` transaction request. In the Capabilities document, `SupportedInputCRS` will be located in `WCS_Capabilities / Capability,` parallel to the `Request` element. It will contain any number of `inputCRS` elements, one for each format the server supports

Implementation Note: The LizardTech WCS-T server does not provide a list of supported input CRSs. It will support any CRS given.

2.3 `DescribeCoverage` Changes

This work entails no changes to the `DescribeCoverage` request or response.

2.4 `GetCoverage` Changes

This work entails no changes to the `GetCoverage` request or response.

2.5 `AddCoverage` Addition

This operation corresponds to the "Add" operation in 06-043.

The `AddCoverage` operation supports these parameters:

- `InputFormat` – file format of input coverage (optional)
- `InputCRS` – CRS of input coverage (optional)
- `InputMetadataURL` – location of input coverage metadata (optional)
- `InputDataURL` – location of input coverage data (optional)

Either HTTP GET or HTTP POST may be used. For POST, `enctype="multitype/form-data"` should be used.

If the URL parameters are used, POST is not necessary. If POST is used, the labels `InputDataBody` and `InputMetadataBody` may be used to identify the corresponding data sections.

The input format and CRS must be in the list of supported input formats and CRSs. If either of these two optional parameters is not specified, the values will be inferred from the file's contents.

The `InputMetadataURL` points to a location providing metadata for the coverage.

The response to an `AddCoverage` request will be the coverage description, or an appropriate error signal if the add operation fails for some reason, e.g. the file format is not supported.

Implementation Note: LizardTech's WCS-T server ignores the `InputMetadataURL` value if it is supplied.

Implementation Note: LizardTech's WCS-T server does not support coverages made up of multiple image files.

2.6 `UpdateCoverageMetadata` Addition

This operation corresponds to the "Update" operation in 06-043.

The `UpdateCoverageMetadata` operation supports these parameters:

- `Coverage` – the ID of the coverage to be updated (required)
- `InputMetadataURL` – location of input coverage metadata (optional)

Either HTTP GET or HTTP POST may used. For POST, `enctype="multitype/form-data"` should be used.

The coverage ID must be a valid ID for the server.

The response should be an acknowledgement of success, or else an appropriate error signal if the update operation fails for some reason, e.g. the coverage ID is not valid.

Implementation Note: LizardTech's WCS-T server does not support this operation.

2.7 `UpdateCoverage` Addition

This operation corresponds to the "Update" operation in 06-043.

The `UpdateCoverage` operation supports these parameters:

- `Coverage` – the ID of the coverage to be updated (required)
- `InputMetadataURL` – location of input coverage metadata (optional)

Either HTTP GET or HTTP POST may used. For POST, `enctype="multitype/form-data"` should be used.

The coverage ID must be a valid ID for the server.

The response should be an acknowledgement of success, or else an appropriate error signal if the update operation fails for some reason, e.g. the coverage ID is not valid.

Implementation Note: In the LizardTech WCS-T implementation, the operation is named `ReplaceCoverage` instead of `UpdateCoverage`.

Implementation Note: In the LizardTech WCS-T implementation, the coverage ID parameter is named `CoverageID` instead of `Coverage`.

Implementation Note: In the LizardTech WCS-T implementation, the new coverage data must match the old coverage data with respect to such properties as width and height, colorspace, bit-depth, etc.

2.8 `DeleteCoverage` Addition

This operation corresponds to the "Delete" operation in 06-043.

The `DeleteCoverage` operation supports these parameters:

- `CoverageID` – the ID of the coverage to be removed (required)

Either HTTP GET or HTTP POST may used. For POST, `enctype="multitype/form-data"` should be used.

The coverage ID must be a valid ID for the server.

The response should be an acknowledgement of success, or else an appropriate error signal if the update operation fails for some reason, e.g. the coverage ID is not valid.

3. Discussion

3.1 Security and Authentication

WCS-T makes no restrictions with respect to security or authentication of clients modifying the server's coverage offerings. Such control is left to other OWS mechanisms (as is the case for WFS-T).

3.2 "Coverage-Based" vs. "Region-Based" Semantics

The extensions in this proposal operate on entire coverage offerings, as opposed to regions within a coverage. That is, only an entire coverage layer may be added or deleted; no support is given for modifying or updating a *subset* of an existing coverage.

In the future, however, it is likely we will want to be able to perform transactions on subsets of a coverage. If so, several issues will need to be addressed:

- Not all file formats used by storage on the back end are readily amenable to updating, e.g. compressed JP2 images. It would be useful for the server to be able to indicate via the capabilities document or a coverage description whether a given coverage is updatable by region.
- The incoming data must be of the same type and format as the existing data, e.g. bit depth, number of bands, etc.
- The incoming data must match the existing data with respect to its coordinate reference system, etc.
- What should the behavior be if you delete part of the interior of a coverage? For example, would a newly-created "hole" appear as a no-data region? Similarly, what happens if you add coverage data such that the geographic origin or bounding box is changed?

3.3 "`Update`" for "Versioning"

It is likely that some workflows will require using the `Update` operation as a mechanism for "versioning" the file. For example, the concept of coverage IDs could be extended to include version numbers; an

update would then really be just an `AddCoverage` of a new version. However, this is likely to require better understanding of expected workflows and would in any case have a significant impact on WCS in general; versioning is out of scope for WCS-T. Guidance should be taken from prior art in other W*S specifications.

3.4 Separation of `Add` and `Update`

`Add` and `Update` are treated as distinct operations within the Capabilities document. This allows a server to support only insertions and not updates, or vice versa, depending on implementation issues or security concerns.

3.5 Lack of Support for Nongeoreferenced Images

It was requested that attention be paid to the question of nongeoreferenced images. We did not do so, however, as WCS 1.0 does not support coverages that are not georeferenced; WCS 1.1 will. (There is also no support for this in GMLJP2 1.0, although the RWG is aware of the issue.)

3.6 Atomicity and Locking

The atomic nature of the "transaction" should be explicitly addressed in a subsequent version of this proposal. Specifically, where there is more than one operation requested, if any operation fails then none of the others are to be committed. (This proposal only allows for one operation per request.)

If providing such commit-or-rollback semantics is not going to be feasible for some server implementations, it may be desirable to provide a mechanism for a server to indicate that it does not support these semantics, and/or provide a client-side mechanism for indicating that a sequence need not be performed atomically.

We also do not support locking in this proposal. Subsequent work involving support for multiple operations per single transaction, or for region-based semantics, might require locking semantics. We would expect using an approach similar to that supported by WFS.

4. Examples

Add (via POST):

```
----
Content-Disposition: form-data; name="SERVICE"

WCS
----
Content-Disposition: form-data; name="ACCEPTVERSIONS"

1.0.0
----
Content-Disposition: form-data; name="REQUEST"

AddCoverage
----
Content-Disposition: form-data; name="InputFormat"

image/jp2
----
Content-Disposition: form-data; name="InputCRS"

EPSG:26912
-----
```

```
Content-Disposition: form-data; name="InputDataBody";
filename="blah.jp2"
Content-Type: image/jp2

<INPUT_DATA_BODY>
----
```

Update (via POST):

```
----
Content-Disposition: form-data; name="SERVICE"

WCS
----
Content-Disposition: form-data; name="ACCEPTVERSIONS"

1.0.0
----
Content-Disposition: form-data; name="REQUEST"

ReplaceCoverage
----
Content-Disposition: form-data; name="COVERAGE"

incoming/cov1E.jp2
----
Content-Disposition: form-data; name="InputDataBody";
filename="blah.jp2"
Content-Type: image/jp2

<INPUT_DATA_BODY>
----
```

Delete (via GET):

```
http://example.com/wcs-server?
SERVICE=WCS&
REQUEST=DeleteCoverage&
COVERAGE=coverageToDelete.jp2
```