Paper for Consideration by S-100 TSM5

Updating GML datasets

Submitted by:	Raphael Malyankar & Eivind Mong
Executive Summary:	Describes an approach to updating GML datasets.
Related Documents:	--
Related Projects:	S-100; all product specifications using the GML data format

Introduction / Background

This paper presents an approach to defining update datasets encoded using the GML format. An "update dataset" is defined as a collection of data that can be applied to an existing dataset to add, delete, or amend specific instances of features, information types, or spatial types. An update dataset as defined in this paper is not a complete self-contained dataset that can replace an existing dataset in its entirety but a delta that is applied to a dataset to produce a new version of the dataset.

The scope of this paper is limited to datasets in the GML format.

References

[1] The IEEE & The Open Group, POSIX.1-2008. IEEE 1003.1-2008/Cor 2-2016 - Standard for Information Technology--Portable Operating System Interface (POSIX®) Base Specifications, Issue 7, also known as The Open Group Technical Standard Base Specifications, Issue 7. <http://pubs.opengroup.org/onlinepubs/9699919799/>.

[2] Urpalainen, J., "An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath) Selectors", RFC 5261, DOI 10.17487/RFC5261, September 2008, <http://www.rfc-editor.org/info/rfc5261>.

Approaches

Possible approaches to updating are listed below, and discussed in more detail in the rest of this section. 'Updates' in the following includes additions, deletions, and replacements.

1) Attribute-level update – update delta dataset describes updates to attributes as well as instances.
2) Object-level update – update delta dataset described updates only at the instance level.
3) Update with a text-based 'patch' utility – update is a purely text-based delta between the current and updated dataset that ignores structure and semantics.
4) XML-aware patching – update is an XML-based delta that identifies the XML item (attribute or instance) being updated.
5) Whole-dataset replacement – the entire dataset is distributed when an update is necessary.

Attribute-level update

This is similar conceptually to the approach described in S-100 Part 10a. Only the new or changed parts of the features/information types/spatial object instance would be included in the update dataset. This approach would need definition of an update delta dataset structure. Since an update to a feature generally does not change all the attributes of the original feature, it will also be necessary to define a new GML schema for update datasets for each product specification which allows object types (features and information types) to contain any of the attributes bound to it in the feature catalogue while relaxing all constraints which make attributes mandatory. Structurally this would be very similar to the application schema that define the base dataset format, but it would relax requirements for mandatory attributes. It will also be necessary to define a format for deltas to spatial types that is capable of indicating exactly which component of the spatial type (curve segment, point, etc.) is being updated.

This approach also includes updates at the instance level, i.e., new feature, information type, and spatial instances can be added, deleted, and replaced.

Object-level update

Only whole instances can be added, deleted, or replaced. Even if the only change is to the value of a single attribute, the whole feature or information type instance type is included in the update dataset. The update dataset contains whole instances of the object being updated. For this approach, it is not necessary to maintain a current version of the dataset in order to generate the delta, only to record the identity and versions of instances in the dataset.

Update with text-based patch utility

Since GML datasets are text files, another possibility is to treat them as ordinary text files and use a "patch" utility [1] to apply changes distributed in the form of a patch file. Text patch utilities are system utilities on Unix and Linux systems and there are open-source implementations available for Windows operating systems. The 'patch' method requires that the file being patched on the end-user system mirror the version on the production system. It also assumes that all predecessor patches have been applied. The update dataset here would consist of a collection of fragments – generally one or more lines – that are being added or replaced, accompanied with location information (such as line numbers and/or surrounding lines for context) that tells the patch utility where in the dataset to apply the patch. No XSD file is required for the update dataset since its format must conform to the 'diff', 'ed', and 'patch' utilities, which format is described in the cited standard.

XML-aware patching

A specification for describing patches to XML documents is available (reference [2]). This specification uses XML Path Language (XPath) selector expressions to locate the XML content being changed and defines basic add, replace, and delete operations. There are open-source utilities that can generate a patch file conforming to this specification and other utilities that can apply the patch file to XML files. This is also conceptually similar to the attribute (and higher) level update approach defined in S-100 Part 10a. The structure of the update dataset is described by the schema defined in the cited reference and is independent of the GML application schema for the specific data product. In this approach it would be necessary to maintain a current version of each object as distributed in order to generate each delta update, but proper definition of the XPath expressions (leveraging the fact that objects have IDs that are unique in the dataset – and also globally unique, if global identifiers are introduced) should reduce or remove altogether the need to maintain a current copy of the dataset in order to generate the next delta update.

An example for an addition of a *restriction* attribute to a **RestrictedAreaNavigational** feature:

```
<add sel="//RestrictedAreaNavigational[@gml:id = 'RESARE55']">
<restriction>entry prohibited</restriction>
</add>
```

Whole-dataset replacement

This method of updating consists of distributing an updated dataset in its entirety when an update is needed. It would be the default method in the absence of any of the others. The replacement would be a new edition. The volume would be highest. For certain kinds of data products and certain application domains this would in fact be the most appropriate method, especially if the data are volatile over most or all of the dataset coverage or if bandwidth limitations are not a factor. Weather images are a case in point. Another example is S-124 navigation warnings.

Analysis of approaches

The approaches are not all mutually exclusive. For example, both attribute- and object-level update can be achieved using a patch file or XML-aware patching, combined with appropriate rules for creating the update dataset. Defining a specification that allows attribute-level update should be feasible using option 4 (XML-aware patching) in the list. The table that follows compares the approaches.

A large proportion of the total size of many datasets is due to long strings of geographic coordinates necessary for encoding curves and surfaces. Replacing them in their entirety will add a significant amount of overhead when only a few coordinates are changed. Similar considerations apply to the values of gridded data or value sets associated with multi-points, such as **Sounding** and **Depth – no bottom found** features in S-101. Special methods can be developed that allow delta updates to contain only the sections of the lists or value sets which are being updated.

Table 1. Comparative analysis of approaches to updating S-100 datasets

Factor/Approach	Attribute-level	Object-level	Text patch	XML patch	Whole-dataset
Update detail	Attribute, instance, spatial object, spatial primitive	Instance, spatial object	Line-based	Attribute, instance, spatial object, spatial primitive	Dataset as a whole
Schema	Modification of base dataset GML schema	Same as base dataset GML schema	Not needed	Special XML schema common to multiple products	Same as base dataset
Robustness	Flexible, can work around some types of errors, sequential updates loosely coupled. Cumulative patch can be created.	More flexible, can work around more types of errors, coupling of sequential updates even looser. Cumulative patch can be created.	Least flexible, tight coupling of sequential updates. Cumulative patch possible but requires original dataset to be available exactly as issued including line breaks.	Comparable to Attribute-level approach.	Robust
Volume of update delta	Low	Greater than Attribute level, but much lower than whole-dataset replacement.	Comparable to Attribute level	Comparable to Attribute-level	High
Tools - availability	OEMs must develop.	OEMs must develop.	System utilities or off-the-shelf, open-source applications. 'Wrappers' must be developed.	Uncertain. Could re-use or adapt open-source tools or develop custom application.	No special functionality
Implementation effort	Highest	Low	Low	Moderate	Low
Application logic	Application logic like 8211 format.	Simpler parts of 8211 logic since attributes are not individually updated.	None for basic implementation. Error recovery depends on S-100 requirements.	Simple logic for basic implementation. Error recovery depends on S-100 requirements.	Simplest
Sotware re-use	Reusable by all data products conforming to S-100 GML schema conventions.	Reusable by all data products conforming to S-100 GML schema conventions.	Re-usable by all GML-encoded data products even if they do not follow S-100 GML conventions.	Reusable by all data products conforming to S-100 GML schema conventions.	Reusable
Update tracking and object history on end-user system	More complex – application must generate and track	Simplest - maintain previous versions of objects	Most complex – application must identify feature, generate and track	Comparable to Attribute-level	Application must manage tracking and history. Difficult except under very limited circumstances (e.g., if a 'dataset' contains only a single feature)

Factor/Approach	Attribute-level	Object-level	Text patch	XML patch	Whole-dataset
Updating spatial type	Update feature spatial attribute as a whole, or separate spatial object. Functionality to update an individual coordinate involves more complex specification and application development.	As part of feature, or as separate spatial object. Shared geometry changes may require updates to all affected features. Special treatment needed for updating individual coordinate.	Dependent on externalities such as line breaking. Special treatment for individual coordinate not possible in general.	Update feature spatial attribute as a whole, or separate spatial object. Functionality to update an individual coordinate involves more complex specification and application development.	No special treatment needed
Previous version requirements – producer	Minimal – can track and generate in production database	Minimal	Must have older version(s) of GML file available for creating delta.	Minimal	Not needed except for product-specific tracking and history purposes.
Previous version requirements – end-user	Minimal (e.g. object being updated must exist, etc.)	Minimal	Must have older version of GML file available and this must match the file on the producer end.	Minimal	Not needed except for product- or application-specific tracking and history needs, if any.
Transfer format	Files. Potentially compatible with WFS servers or custom S-100 web services.	Files. Compatible with WFS servers or custom S-100 web services.	Files. Not compatible with Web services.	Files. Less compatible with web services than Attribute-level.	(Same as base dataset, so it depends on how the product specification distributes data.)
Data product types	Appropriate for data products built on discrete feature-based or information types	Appropriate for data products built on discrete feature-based or information types	Appropriate for data products built on discrete feature-based or information types	Appropriate for data products built on discrete feature-based or information types	Appropriate for imagery data products, gridded data, weather images, single-feature-datasets (S-124 nav warnings)
Compatibility with GML 3.2.1	Possible – but will involve using metadata or special attribute to specify operation (insert / delete / update).	Yes	Yes - ignores GML altogether, uses physical position in file, or neighbouring content in dataset.	Yes - update itself will use a non-GML schema but the method can reference items in any GML file.	Yes

Taking the above analysis into account, this paper recommends that S-100 be extended with the following two approaches for updating GML datasets:

- Object-level updating for products based on discrete features or information types, subject to two caveats:
    - Products where datasets are very small will generally need only whole-dataset methods; and
    - Data or coordinate collections or sequences will need special treatment (e.g., functionality extensions to locate coordinates by their positions in the sequence) in order to reduce the average data volume of updates.
- Whole-dataset updating for imagery, gridded data, etc., with the caveat below:
    - Methods for updating parts of gridded or pointset data in other formats should be adapted to GML encodings.

Exceptional circumstances should be considered when making the decision as to which method(s) a data product uses, for example whether datasets contain only one instance or whether all the instances are associated (e.g., a feature instance associated to some information type instances), the two approaches are almost equivalent and the whole-dataset method can be used since it is simpler. Similarly, if it is expected that the majority of the data within the dataset will normally require updating, then whole dataset replacement (e.g., as a new edition) should be considered – this last decision would be made by the producer and be independent of the update approach chosen by the product specification, since product specifications will normally permit re-issues and/or new editions. For simplicity, product specification authors are generally expected (though not mandated) to pick one approach for updating datasets, keeping in mind the permissibility of new editions and re-issues.

The remainder of this paper describes the requirements for object-level updating, update structure, and how product specifications should be written to implement this method.

Detailed description

Update unit

The units of update are 'whole objects'. A 'whole object' for the purposes of this paper is any instance of a feature, information type, or spatial type in an S-100 dataset. Note that an 'instance of a spatial type' means an independent spatial object that is encoded independently of any feature instance (i.e., not embedded within a feature instance).

The feature ID attribute for a feature being modified must be the same as the data object being replaced. The *gml:id* attribute of the replacement should be the same as that of the original - this is to avoid the need to update every reference to the object, since references use the gml:id values. The requirement that the *gml:id* be the same implies that original and replacement cannot both be simultaneously present in a dataset.

An alternative to requiring re-use of *gml:id* for modified versions is for the specification for the update process to require updating of references in other objects in the dataset when the update is loaded. A second alternative is to define internal structure for *gml:id* values (e.g., ending with ".0", ".1", etc. to indicate successive versions). Both alternatives imply some new constraints on GML application schemas and extensions to their semantics for S-100 datasets (e.g., the interpretation and use of *gml:id* values for referencing objects).

Version numbers

To facilitate update management and tracking, a record version field similar to RVER in the ISO 8211 encoding should be added to feature, information type, and spatial objects; for standardization across all product specifications, this should be done in the S-100 GML profile.

GML format for update datasets

The format for update datasets is the same as for base datasets. A replacement feature instance, information type instance, or spatial object will have the same *gml:id* XML attribute as the instance it replaces – GML applications can still distinguish between original and replacement by using the update dataset file name as a prefix to the *gml:id* value.

In general the data format defined for the base dataset should be re-usable for the update delta. For volume-reducing methods, there may be some special encoding rules, similar to the 'empty container' encoding example provided later in this section.

If the base dataset structure prescribes an ordering of types, the update file structure should follow the same ordering, for the same reasons. Updated objects of the same type should be ordered according to the order of the original objects in the dataset to allow a (potentially) faster process for application of the update – this is not mandatory and production tools may deviate from this recommendation if necessary.

With the 'whole object' paradigm and re-use of feature ID and/or *gml:id*, insertions and modifications do not need to be explicitly indicated as such, because the update process can in principle test for the existence of an object (using the re-used feature ID or *gml:id*) and treat the update as a modification if it is found and an insertion if it is not found.

The data format for inserted and modified features is the same as the data format for originals, e.g.:

```
<imember>
  <S122:Recommendations gml:id="USRCMDTS99">
    <textContent>
      <categoryOfText>full text</categoryOfText>
      <information>
        <headline>Avoidance of whale pods</headline>
        <language>eng</language>
      </information>
      <onlineResource>
        <linkage>http://www.noaa.gov/whale/pods/current/location</linkage>
         <nameOfResource>Whale pod information</nameOfResource>
         <onlineDescription>Whale pod avoidance recommendations</onlineDescription>
      </onlineResource>
    </textContent>
    <appliesInLocation xlink:href="#USSEAARE1"
          xlink:role="http://www.iho.int/S-122/gml/1.0/roles/appliesInLocation"/>
  </S122:Recommendations>
</imember>
```

Feature and information type instances are deleted without replacement by setting the *fixedDateRange.dateEnd* attribute of the instance to the date of deletion, which will usually be the issue date of the update. As a more compact method, the object container may be nilled with an appropriate *nilReason*, but this lacks the 'history' aspect of setting the *dateEnd* attribute to indicate the lifetime of the feature and requires that GML application schemas allow such empty containers. Examples of both are shown below.

```
<!-- Method 1 – empty container with nil reason -->
<member xlink:href="USMPAARE5" xlink:title="Delete feature USMPAARE5"
        nilReason="other: deleted" />

<!-- Method 2, update the fixedDateRange.dateEnd attribute -->
<member>
  <S122:MarineProtectedArea gml:id="USMPAARE5">
    <fixedDateRange>
     <dateEnd>
        <date>2017-05-04</date>
      </dateEnd>
    </fixedDateRange>
    etc., etc.
  <S122:MarineProtectedArea>
</member>
```

Operations must be applied to the most current version of the dataset, i.e., all prior updates must have been applied in order.

Dataset cancellation (termination)

In order to cancel a dataset, an update dataset file is created for which the edition number must be set to 0. This message is only used to cancel a base dataset file.

Where a dataset is cancelled and its name is reused at a later date, the issue date must be greater than the issue date of the cancelled dataset.

When a dataset is cancelled it must be removed from the system.

Dataset structure

The update dataset is fundamentally a dataset that conforms to the same GML schema as the base dataset, contains feature, information, and spatial object instances being updated (including additions and deletions). The differences are:

- Requirements pertaining to the presence of meta-features do not apply to the update delta dataset. Meta-features are included in the update delta only if they themselves are being updated. After the update is applied, any rules concerning meta-features of course apply to the resultant dataset.
- Constraints pertaining to the presence of associated features and information types do not apply to the update delta dataset. Associated features and information types are included only if they themselves are being updated. After the update is applied, any constraints applying to associations of course apply to the resultant dataset.
- Similarly, rules pertaining to the presence of referenced spatial objects or referencing features do not apply to the update delta dataset , but do apply to the resultant dataset as usual.

Dataset types

GML data products can include the same types of datasets within an exchange set as defined for the ISO 8211 encoding:

Table 2. Dataset types

Dataset	Explanations
New dataset (base dataset)	Data for an area different (in coverage and/or extent) to existing datasets.
New Edition of a dataset:	A re-issue plus new information which has not been previously distributed by Updates. Each New Edition of a dataset must have the same name as the dataset that it replaces and should have the same spatial extents.
Update dataset	A delta change of the latest edition of a dataset. If there are more than one update dataset, the subsequent update will be a delta of the base dataset + earlier update datasets.
Re-issue	Complete dataset including all the updates applied to the original dataset up to the date of the reissue. A re-issue does not contain any new information additional to that previously issued by updates.
Termination	Used to terminate dataset. Some product specifications use the term “Cancellation.”

Data coverage

An update dataset must not change the limit of a Data Coverage feature for the base dataset. Where the limit of a Data Coverage feature for a base dataset is to be changed, this must be done by issuing a new edition of the dataset.

Dataset loading

Datasets must always be loaded in the order of base dataset first, then update datasets in the correct sequential order. Systems are not to load updates out of order, for example if update 1-5 are present, then 6 is missing, update 7 must not be loaded.

Size

Reasonable limits on the size of update datasets may be imposed. E.g., if the base dataset limit is 10MB, update datasets may be limited to a size of at most 500KB. The nature of the product domain must be considered when setting size limits, e.g., some domains may have updates that are nearly as large as base datasets.

File naming

All update dataset files should have an identical name to the base dataset, aside from the separator and update number sequence.

Reuse of names of cancelled datasets

Where a dataset is cancelled and its name is reused at a later date, the issue date must be greater than the issue date of the cancelled dataset.

Exchange set contents

An exchange set may contain base dataset files and update dataset files for the same datasets. Under these circumstances the update dataset files must follow in the correct sequential order from the last update applied to the base dataset file. Product specifications should describe how systems should behave if the update sequence is broken.

Support file updates

The purpose of issue is indicated in the "purpose" field of the support file discovery metadata. Support files carrying the "deletion" flag in metadata must be removed from the system. When a feature or information type pointing to a text, picture or application file is deleted or updated so that it no longer references the file, the system software must check to see whether any other feature or information type references the same file, before that file is deleted.

Updates or deletions of a support file may require concurrent updates to feature or information type instance attributes that depend on the file, e.g., *pictorialRepresentation*, *fileReference* and *fileLocator* attributes.

Validation checks

Validation checks for base datasets should be reviewed and – for the purpose of validating updates – classified as being applicable to the update in isolation, or to the entire dataset after updating. This classification depends on whether the condition references other features or information types in the whole dataset which may not have been included in the update. The validation checks part of the specification must specify which tests are applied to the base dataset, which to an update in isolation, and which to the updated dataset. Some checks may be applied in multiple sets of circumstances. A few new tests applying only to updates will need to be added. The table below contains examples from S-122.

Table 3. Sample validation checks showing applicability to base datasets (B), Updates only (U), and post-update whole datasets (S)

Check description	Check message	Check solution	Conformity to:	Apply to
For each feature object where its geometry is not COVERED_BY a DataCoverage	Objects fall outside the coverage object.	Ensure objects are not outside of the limits of the cell.	PS 8.9	B, S
For each feature record where the name is not unique WITHIN the dataset.	Duplicate FOIDs exist within the dataset.	Ensure that no duplicate FOIDs exist.	PS 8.8	B, U, S
If any mandatory attributes are not Present.	Mandatory attributes are not encoded	Populate mandatory attributes (If unknown encode attribute with empty value).	DCEG 2.4.2	B, U
For each instance of ServiceHours has more than one instance of scheduleByDoW, and where an instance of scheduleByDoW has a temporal overlap with another instance of scheduleByDoW.	Schedule overlaps	Review service hour intervals and remove time overlap.	Logical consistency	B, U
If the update dataset file size is greater than 500KBytes	The update is larger than 500 KB in size.	Ensure that the cell is not larger than 500 KBytes	PS 8.11.3	U

Metadata

Updates to metadata

Base dataset discovery metadata cannot be changed by an update dataset. Other metadata e.g., for support files might be changed as necessary, but the process and considerations for updating metadata would be separate from datasets though the possible approaches will probably be similar to those analyzed in this paper and minor changes to the metadata schemas may be necessary (e.g., XML ID attributes for metadata elements).

Metadata for update datasets

Update dataset metadata is intended to describe information about an update dataset. It facilitates the management and exploitation of data and is an important requirement for understanding the characteristics of an update dataset. Whereas dataset metadata is usually fairly comprehensive, metadata for update datasets only describe the issue date and sequential relation to the base dataset.

The purpose of issue of the dataset is indicated in the "purpose" field of the dataset discovery metadata. In order to terminate a dataset, an update dataset file is created for which the edition number must be set to 0. This convention is only used to cancel a base dataset file.

Table 4. Discovery metadata for update datasets

Name	Cardinality	Value	Type	Remarks
S100_DatasetDiscoveryMetadata				Discovery metadata for the update dataset
fileName	1		CharacterString	Update Dataset file name
filePath	1		CharacterString	Full path from the exchange set root directory
description	1		CharacterString	Brief description of the update.
dataProtection	0..1	{1} or {2}	CharacterString	Value must be same as base dataset.
protectionScheme	0..1		CharacterString	Value must be same as base dataset.
digitalSignature	0..1		CharacterString	If the base dataset is signed the update should normally also be signed.
digitalSignatureReference	0..1		CharacterString	As for base dataset
digitalSignatureValue	0..1		CharacterString	As generated for the update
copyright	0..1		MD_LegalConstraints ->MD_RestrictionCode <copyright> (ISO 19115)	Value must be same as base dataset.
classification	0..1		Class MD_SecurityConstraints>MD_ClassificationCode (codelist)	Value must be same as base dataset.
purpose	1	{3}, {4}	CharacterString	3. Update 4. Terminated (or Cancellation)
specificUsage	1		MD_USAGE>specificUsage (character string) MD_USAGE>userContactInfo (CI_ResponsibleParty)	brief description of the resource and/or resource series usage. Same as base dataset.
editionNumber	1	{1}	Integer	Value must be same as base dataset.
updateNumber	1		CharacterString	Update sequence number, must match file name.
updateApplicationDate	1		Date	Date of update
issueDate	1		Date	Date on which this update was issued.
productSpecification	1		CharacterString	Value must be same as base dataset.
producingAgency	1		CI_ResponsibleParty	Party responsible for generating the update.
horizontalDatum	1		CharacterString	The datum for latitude/longitude. Same as base dataset (usually EPSG:4326)
verticalDatum	1		CharacterString	Same as base dataset (usually EPSG:4326)
dataType	1	GML	CharacterString	

dataTypeVersion	1	3.2.1	CharacterString	
dataCoverage	1..*		S100_DataCoverage	Same as base dataset
comment	0..1		CharacterString	Any additional Information
layerID	1..*		CharacterString	Value must be same as base dataset. E.g., S-101

Table 5. Metadata for updated support files

Name	Cardinality	Value	Type	Remarks
S100_SupportFileDiscoveryMetadata				
fileName	1		CharacterString	
fileLocation	1		CharacterString	Path relative to the root directory of the exchange set. The location of the file after the exchange set is unpacked into directory <EXCH_ROOT> will be <EXCH_ROOT>/<filePath>/<filename>
purpose	1		S100_SupportFilePurpose	new, replacement, or deletion
editionNumber	1		CharacterString	
issueDate	1		Date	
productSpecification	1		S100_ProductSpecification	May differ from original, but must be consistent with the type of support file (as updated). E.g., it is possible to change from JPG to TIFF (assuming both are allowed in the product specification).
dataType	1		S100_SupportFileFormat	May differ from original, but must be consistent with the reference to the support file (as updated). Any pointers to locations within text support files must be updated simultaneously. E.g., a support file referenced by attribute pictorialRepresentation must be a graphic file of a type allowed by the product specification.
otherDataTypeDescription	0..1		CharacterString	
dataTypeVersion	1		CharacterString	
comment	0..1		CharacterString	
digitalSignatureReference	0..1		CharacterString	Reference to the appropriate digital signature algorithm
digitalSignatureValue	0..1		CharacterString	

Conclusions

The possible approaches to defining an update format for GML datasets has been analysed. The most promising are the 'whole-object' and 'XML patch' approaches. The 'whole-object' approach offers a reasonable compromise between the various factors and should be evaluated in S-100 test-beds. The 'XML patch' approach is another candidate but may need more development of format details and update processing before it can be implemented.

Recommendations

- Evaluate the most promising approaches to an update format for GML in S-100 testbeds, especially the 'whole-object' approach. The 'XML patch' approach may be considered as another candidate.
- Develop rules or guidelines for update format, processing, and management for eventual incorporation into S-100 or supporting guidance for authors of product specifications and implementers.

Action Requested of S-100 WG TSM

The TSM is invited to:

a. Discuss this paper.

b. Select one or more approaches for further investigations in test-beds and more detailed development specifications.

c. Recommend the selected approach(es) for consideration by the full S-100 working group and testing in S-100 testbeds.