

The Parkfield/Landers Reference Earthquakes Digital Library

Brad Aagaard (USGS Menlo Park)
Greg Beroza (Stanford University)
Alexei Czeskis (Purdue University, SCEC summer intern)
Jessica Murray (USGS Menlo Park)
Anupama Venkataraman (Stanford University)

September 9, 2005

1 Reference Earthquakes Digital Library

The reference earthquakes digital library (<http://www.wr.usgs.gov>) is designed with the objective of archiving models associated with earthquakes in standard formats with associated metadata in a common, permanent repository in order to facilitate searching, retrieval, and comparison of models. The library is currently setup to archive models from the 1992 Landers and 2004 Parkfield earthquakes. The idea of a reference earthquake stems from the need to combine results from multiple disciplines to understand the earthquake process. It is particularly important to provide curation and ready access to earthquake models because their lifespan is otherwise extremely limited, often depending on postings to personal web-pages. Current practice often leads to the loss of models or to the absence of sufficient documentation and file format information to render models useful.

The digital library includes an archive of the models tagged with metadata, a user-friendly interface for submitting and updating models, and interfaces for browsing, searching, and downloading archived models, as well as for assessing the extent and variety of data available for each event. The digital library was built from the tools developed by the San Diego Supercomputer Center.

With input from members of the earthquake research community we decided on the metadata and model formats. The digital library will archive metadata and model files for

1. Rupture models,
2. Seismicity catalogs,
3. Surface rupture,
4. Focal mechanism catalogs,
5. GPS displacements,
6. Stress change calculations, and
7. Surface deformation (InSAR, LIDAR),

and metadata for

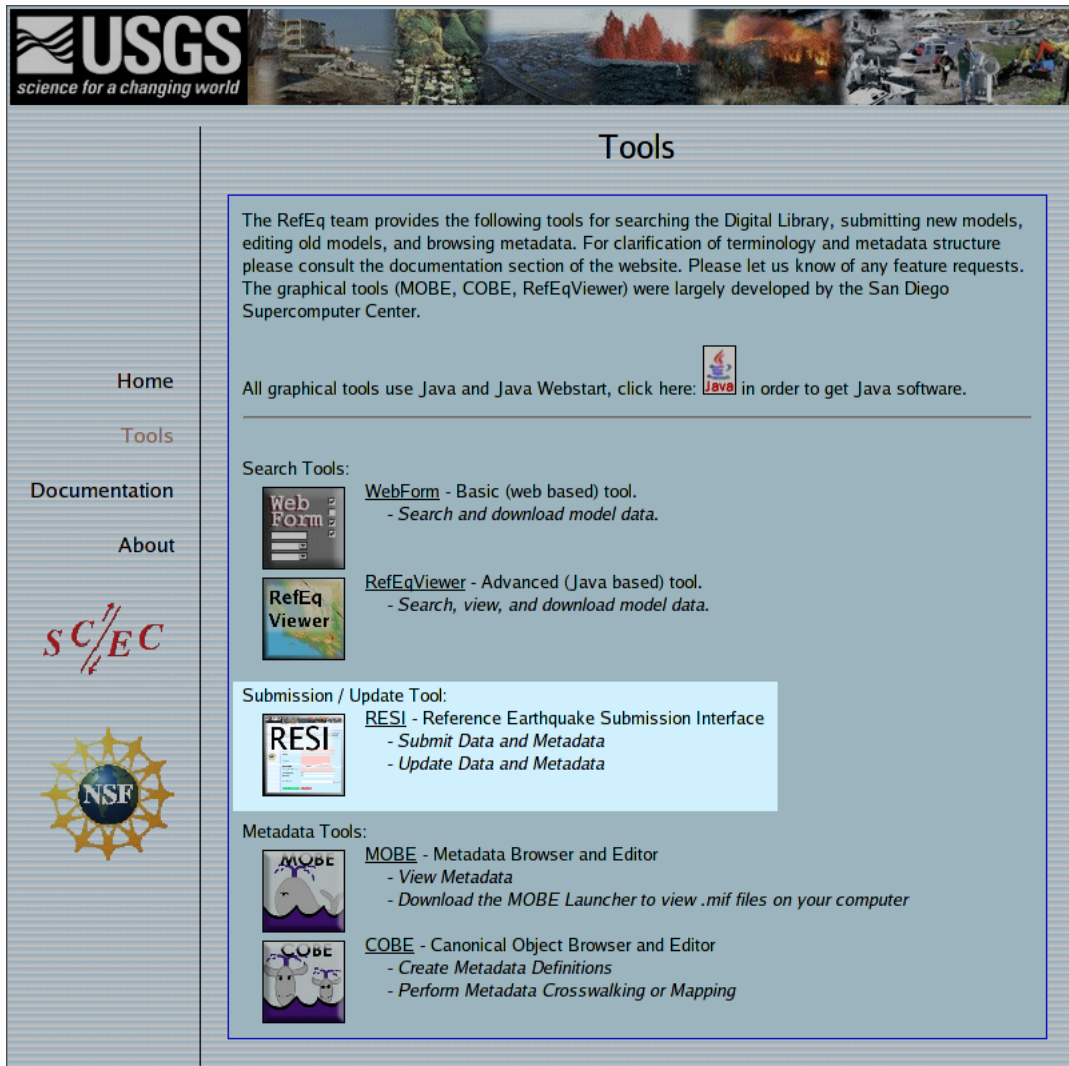
1. Seismic instrumentation and
2. Geodetic instrumentation.

Currently, the digital library can accept rupture models, seismicity catalogs, and surface rupture measurements. Seismic and geodetic instrumentation information has been uploaded into the digital library by the administrator and contains information on some of the seismic and geodetic instruments that were active in the region and during the particular reference earthquake. The remaining instrumentation will be added in the near future.

1.1 Submitting Models

Each of the reference earthquake models has two components: metadata and data. A user who is submitting a model will use the web submission interface RESI (Reference Earthquakes Submission Interface), and follow the submission instructions to submit all the metadata associated with the model. The steps involved in submission are detailed below:


1. To access RESI, click on the RESI icon in the tools section of the Reference Earthquake Digital Library website. (figure 1.1).
2. Select the operation, reference earthquake, and model type.
 - (a) Under what would you like to do? choose Submit a new model.





USGS
science for a changing world

Tools



The RefEq team provides the following tools for searching the Digital Library, submitting new models, editing old models, and browsing metadata. For clarification of terminology and metadata structure please consult the documentation section of the website. Please let us know of any feature requests. The graphical tools (MOBE, COBE, RefEqViewer) were largely developed by the San Diego Supercomputer Center.

All graphical tools use Java and Java Webstart, click here:  in order to get Java software.


[Home](#)
[Tools](#)
[Documentation](#)
[About](#)

Search Tools:

-  [WebForm](#) - Basic (web based) tool.
- Search and download model data.
-  [RefEq Viewer](#) - Advanced (Java based) tool.
- Search, view, and download model data.

Submission / Update Tool:

-  [RESI](#) - Reference Earthquake Submission Interface
- Submit Data and Metadata
- Update Data and Metadata

Metadata Tools:



-  [MOBE](#) - Metadata Browser and Editor
- View Metadata
- Download the MOBE Launcher to view .mif files on your computer
-  [COBE](#) - Canonical Object Browser and Editor
- Create Metadata Definitions
- Perform Metadata Crosswalking or Mapping

Figure 1.1: The RESI (Reference Earthquake Submission Interface) tool is used to submit and update models.

Figure 1.2: RESI front page.

- (b) Under `What Collection?` choose the appropriate reference earthquake.
 - (c) Under `What model would you like to do this to?` choose the appropriate model type.
 - (d) After choosing all of the appropriate choices, click `I'm ready, Let's go!`.
3. Enter metadata and specify the model data file.

Once the above information is submitted, RESI will generate a form with metadata sections that are required by the model you chose to submit. Figure 1.3 gives an example for a seismicity catalog.

This form always contains the General Information and Geographic Extent sections. The form will also contain the core metadata components for the model and other metadata components associated with the model type, such as Software. Section 2 contains more information about the metadata components.

The Keywords and Coverage (and some other) fields permit multiple selections. In order to choose more than one item, hold down the CONTROL key while at the same time pressing the left mouse button over the items that you wish to select.

If a field does not apply to a model, N/A, should be written. Blank fields are not allowed and will not pass the form validation (as discussed later).

Some fields will already have values. Those values show the format of the required input. For example, a field having a value YYYY-MM-DD expects a date, where YYYY is the year, MM is the month, and DD is the day.

Moving the mouse over the name of the field will create a small popup with a description of what is expected for that metadata field.

The model data file is specified at the bottom of the form. Click on the `Browse` button to use a file browser to find the model data file on your computer.

Once you have filled in all of the metadata and specified a model data file, click on the `Submit Information` button.

4. Validation of metadata and model data file.

USGS
science for a changing world

Tools > RESI > Enter Data > Confirm Data

Enter Metadata and Upload Data

Mandatory fields are shaded. Please do not leave mandatory fields blank. If it does not apply to your submission, please write "N/A". Please have the javascript on your web browser enabled. This will allow you to move your mouse over the field names and see description of the terms. Javascript is also used to validate this form.

General Information

Title:

Submitter:

Authors:

Affiliations:

End Date:

Magnitude Min:

Magnitude Max:

Velocity Model Description:

Velocity Model ID:

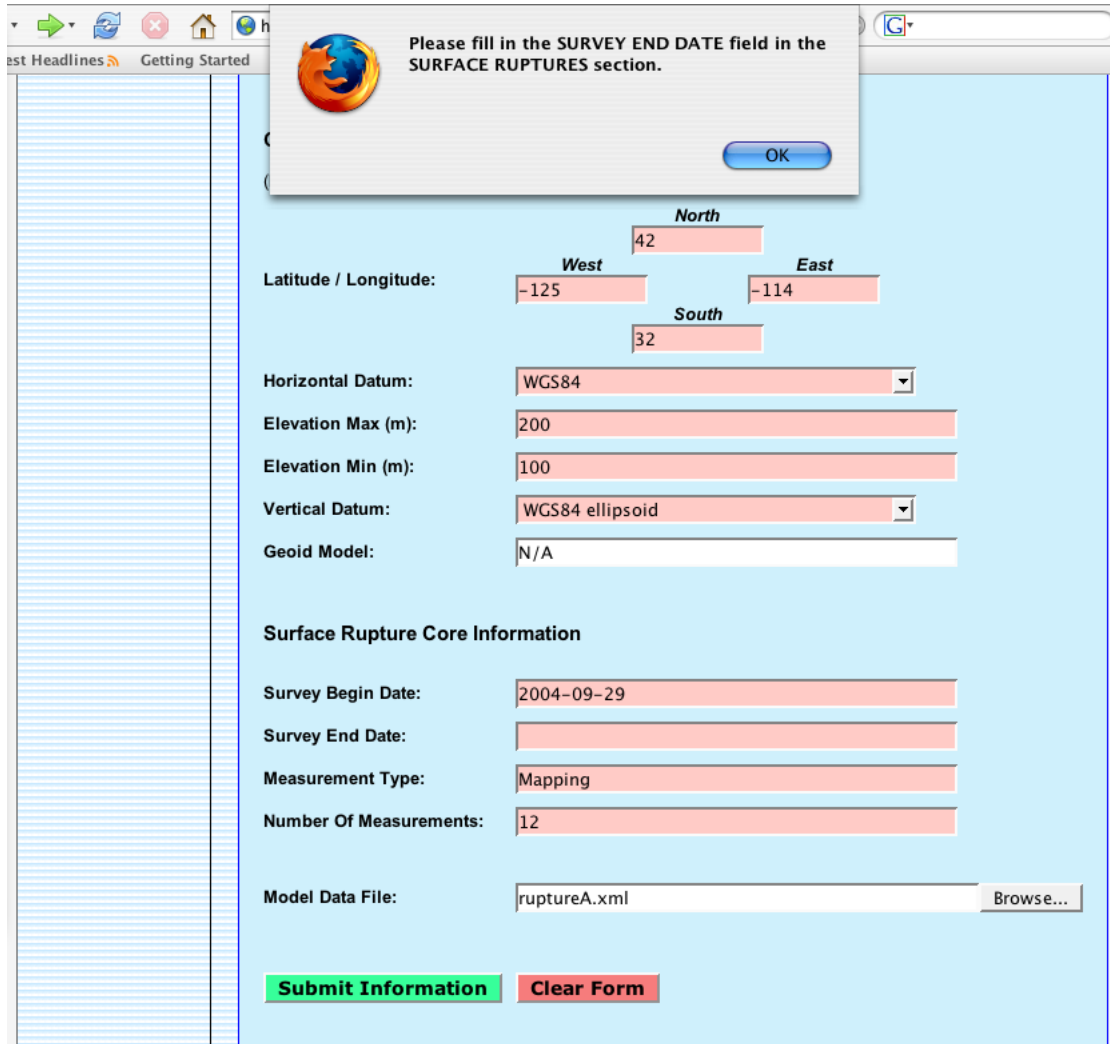
Availability:

Model Data File:

Figure 1.3: The top and bottom portions of the RESI form for entering model metadata and data file.

Upon submission, the metadata and model data file will be validated. The metadata values will be checked to make sure they are in the proper form and then the format of the model data file will be checked to make sure it conforms to the format specifications for the model type you selected.

If the metadata does not pass the form validation, a dialog box will popup displaying the error associated with the metadata (see figure 1.4).



The image shows a web browser window displaying a form for metadata and model data submission. A dialog box is overlaid on the form, indicating a validation error. The dialog box contains the text: "Please fill in the SURVEY END DATE field in the SURFACE RUPTURES section." and an "OK" button. The form fields are as follows:

Latitude / Longitude:	West: -125	North: 42	East: -114
		South: 32	
Horizontal Datum:	WGS84		
Elevation Max (m):	200		
Elevation Min (m):	100		
Vertical Datum:	WGS84 ellipsoid		
Geoid Model:	N/A		
Surface Rupture Core Information			
Survey Begin Date:	2004-09-29		
Survey End Date:			
Measurement Type:	Mapping		
Number Of Measurements:	12		
Model Data File:	ruptureA.xml		Browse...

At the bottom of the form, there are two buttons: "Submit Information" (green) and "Clear Form" (red).

Figure 1.4: Example of metadata validation error message displayed in dialog box.



If the data passes the form validation, the user will be taken to the confirmation page.

5. Confirm metadata and model data file submission.

The confirmation page will display all previously entered metadata and the results of the model data file validation. If the model data file passed the validation, you will be asked to check the data for accuracy and given the opportunity to submit the model (see figure 1.5).

If the model data file failed to pass the validation, you will be asked to fix the errors listed at the bottom of the page in the Data File box (see figure 1.6 and section 1.3).

6. After clicking on the Everything is okay button, you will receive confirmation that the model has been submitted (see figure 1.7). The uploader daemon adds new models to the digital library on a regular basis (generally every 10 minutes).

Tools > RESI > Enter Data > Confirm Data

Confirm Data

Please check the following for accuracy. If you want to make changes, use the **BACK** button on your browser.

General Information

Collection	→	PARKFIELD2004
Title	→	Relocated Seismicity Catalog
Submitter	→	John Doe
Authors	→	John Doe
Affiliations	→	US Geological Survey
Keywords	→	3D earthquake relocations travel times
Description	→	...
Velocity Model ID	→	N/A
Availability	→	N/A

Data File

Model Data File	→	catalogA.xml
File size	→	8.0K

File was validated successfully!

If there is an error in any of the metadata above, or you want to submit a different data file, please use the **BACK** button on your browser in order to fix any discrepancies. If you are satisfied please press the button below to submit your **hard work**.

**Everything
is
Correct!**

Figure 1.5: Top and bottom of the RESI confirmation page when the model data file validates successfully.

The screenshot shows the top and bottom portions of a web page titled "Confirm Data". The page header includes the USGS logo and a navigation breadcrumb: "Tools > RESI > Enter Data > Confirm Data". A left sidebar contains navigation links: "Home", "Tools", "Documentation", and "About", along with logos for "SC/EC" and "NSF".

The main content area displays a red error message: "There were errors validating your data file, please see the Data section for details." Below this, a text block instructs the user to check the following for accuracy and use the "BACK" button. The page is divided into two sections by dashed blue borders:

- General Information:** A table listing metadata for the data collection.

Collection	→	PARKFIELD2004
Title	→	Observations of surface rupture
Submitter	→	John Doe
Authors	→	John Doe
Affiliations	→	US Geological Survey
Keywords	→	crack measurement offset feature
Description	→	Measurements of surface cracking from field mapping.
Number of Measurements	→	12
- Data File:** A section showing file details and a validation error.

Model Data File	→	ruptureA.xml
File size	→	8.0K

File could not be validated because of the following errors:

 - [Error] ruptureA.xml:10:19: cvc-complex-type.2.4.a: Invalid content was found starting with element 'observer_A'. One of {'"":observer'} is expected.

Please fix these problems and try again.

Figure 1.6: Top and bottom of the RESI confirmation page when the model data file validation fails.

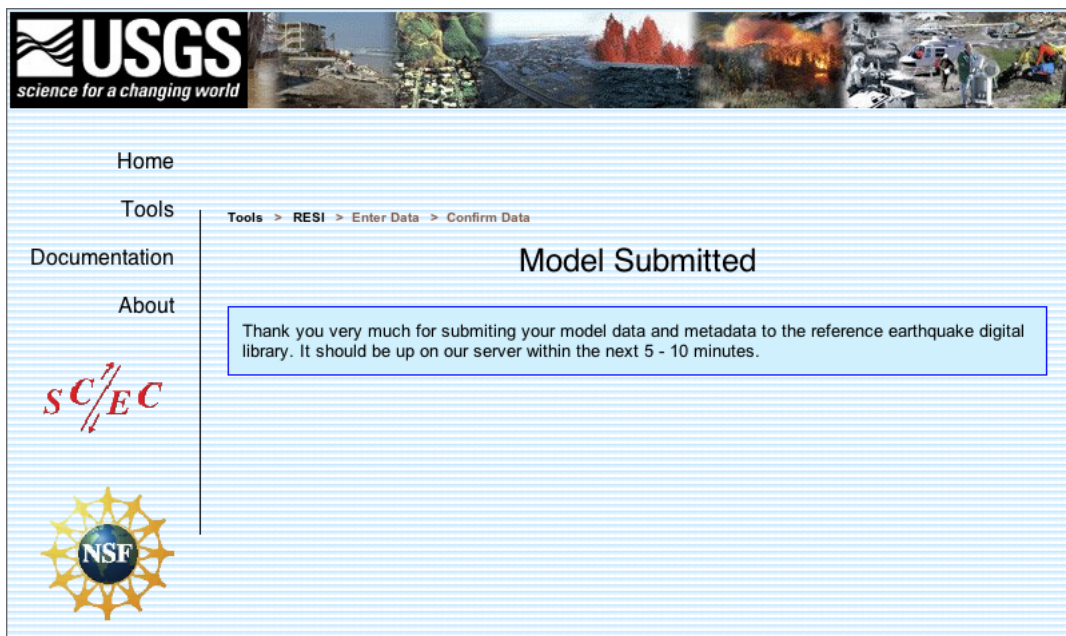


Figure 1.7: RESI model submission confirmation page.

1.2 Updating Models

A model can be updated by editing the metadata and/or submitting a new model data file. Only the digital library administrator can remove models, and this will be done only in the case of major errors. The updated model will be stored as a new metadata and model data file pair; if the model data file is not updated, the previous model data file will be resubmitted with the modified metadata. Updating a model uses RESI and the procedure is very similar to that of submitting a model.

1. To access RESI, click on the RESI icon in the tools section of the Reference Earthquake Digital Library website. (figure 1.1).
2. Select the operation, reference earthquake, and model type.
 - (a) Under What would you like to do? choose Update an existing model (see figure 1.2).
 - (b) Under What Collection? choose the appropriate reference earthquake (see figure 1.2).
 - (c) Under What model would you like to do this to? choose the appropriate model type (see figure 1.2).
 - (d) After choosing all of the appropriate choices, click I'm ready, Let's go! (see figure 1.2).

3. Find the model to update.

Unlike in submitting a model, the page will offer a search form instead of a submission form. This search form should be used to find the model desired for updating (see figure 1.8).

The string (all) can be used as a wildcard in the search form. When the fields are set to limit the search as desired, click on the Submit Information button.

4. Select the model to update.

The search will only look for models of type selected on the RESI start page. If models are found, the title, authors, submitter, and version information will be displayed. A radio button will appear to the left of each model, and you must select which model you would like to update (see figure 1.9). After the radio button for the desired model is selected, click on the Update Selected Model button.

Figure 1.8: The search form for finding the model to update. This particular search returned all of the models submitted by John Doe.

Figure 1.9: Selecting a model to update based on search results.

5. Edit metadata and specify the model data file.

RESI will generate a form with the metadata sections and fill in the values using the values from the model to be updated (see figure 1.10). The user should make changes wherever appropriate.

USGS science for a changing world

Tools > RESI > Find Model > Select Model > Edit Info > Confirm and Submit

Edit Model Metadata

Please make any changes you see fit to the metadata in the form. Please do NOT forget to indicate which type of change this is and note what is new in this version. Also, if you need to update the model data, please do so at the bottom of the page. After you are finished, click on the button at the bottom to continue on to metadata (and data) validation.

General Information

Title: Relocated Seismicity Catalog

Submitter: John Doe

Authors: John Doe

Affiliations: US Geological Survey

Keywords: 1D

Magnitude Max: 6.0

Velocity Model Description: 3-D velocity model from John Smith.

Velocity Model ID: N/A

Availability: N/A

Data

Model Data File: Keep Previous Data Upload New Data Browse...

Apply Changes

Figure 1.10: The top and bottom portions of the RESI form for updating model metadata and the data file.

The version information metadata value requires significant attention. The version number is of the form `major.minor.patch` as in the case of many computer software releases. The digital library keeps track of the version number based on model updates. The user performing the update must explain why the model is being updated and mark this change as a major change, minor change, or a patch. Major and minor versions correspond to major or minor changes in the methodology or datasets used in the model. Patch versions refer to corrections of the metadata or model data file.

At the bottom of the form you can either choose to keep the previously submitted data file or upload a new data file.

After making sure that everything is correct, click on the `Apply Changes` button.

6. Confirm metadata and model data file submission.

The confirmation page will display the current metadata. If a new model data file was used, it will also display the results of the model data file validation. If a new model data file was not submitted, the confirmation page will indicate that the data file from the previous version will be used as shown in figure 1.11.

The screenshot shows the top and bottom of the RESI confirmation page. The top part includes the USGS logo and a navigation breadcrumb: Tools > RESI > Find Model > Select Model > Edit Info > Confirm and Submit. The main heading is 'Confirm Update'. Below this is a message: 'Please check the following for accuracy. If you want to make changes, use the **BACK** button on your browser.' The 'General Information' section lists metadata: Collection (PARKFIELD2004), Title (Relocated Seismicity Catalog), Submitter (John Doe), Authors (John Doe), Affiliations (US Geological Survey), and Keywords (3D, earthquake relocations, travel times, short-period data, and hand data). The 'Data File' section contains a red message: 'You've elected to keep the data file from the previous version (1.1.0)'. At the bottom, there is a green button that says 'Everything is Correct!'.

Figure 1.11: Top and bottom of the RESI confirmation page for an updated model where only the metadata was changed (a new model data file was not submitted).

If a new model data file was submitted and it failed to pass the validation, you will be asked to fix the errors listed at the bottom of the page in the Data File box (see figure 1.6 and section 1.3).

- After clicking on the *Everything is okay* button, you will receive confirmation that the model has been submitted (see figure 1.7). The uploader daemon adds new models to the digital library on a regular basis (generally every 10 minutes).

1.3 Data File Validation

The format of each data file is checked to make sure it conforms to the specified format of data files for that model type. The XML files are checked against an XML schema, while HDF5 files are checked using a C program. Errors found by the validators will be reported at the bottom of the confirmation page.

The XML validator error message reports begin with `[Error] filename:line:column`, where filename is the name of the XML file and line and column refer to the line and column where the error was detected. The validator is generally not sophisticated enough to continue parsing the file after an error is detected, so even though multiple errors may be reported they are usually associated with a single error (e.g., misspelled element name).

The HDF5 validator uses the HDF5 library to attempt to read the HDF5 file. The C program generally dumps helpful error messages. If the error is originally triggered in a routine in the HDF5 library, the validator error messages will begin with a back trace dumped by the HDF5 library. In such cases, it is helpful to begin reading the HDF5 validator error report at the bottom where the more instructive C program error messages will be located.

1.4 Retrieving Models

There are two main ways of retrieving data from the Reference Earthquake Digital Library. One way is through a web interface, the other is through the Java Network Launching Protocol (JNLP). The web form provides quick access to model data files when you know how to narrow a search. The JNLP interface provides graphical tools for searching and downloading.

1.4.1 Webform Searching

Web searching is done through a web form. The user is presented with basic fields describing the models, such as the reference earthquake, model type, model title, authors, keywords, and geographic bounding box (see figure 1.12). The default values should return all the contents of the collection selected. After you fill in the fields as desired, press the `Search!` button. The digital library will search for models that satisfy your criteria.

The search results will be shown on a new page. The title, reference earthquake, file size, bounding box, authors, and version information for each model will be displayed (see figure 1.13). The title is a link to the Arbitrary Digital Object (tar file containing the model data file(s)), and the metadata is a link to the metadata file (MIF). The metadata file is a text file that is most conveniently viewed in the Metadata Browser and Editor (MOBE) (available from the tools page).

1.4.2 RefEqViewer

The RefEqViewer software, written in JAVA and distributed via JNLP, integrates a map of the reference earthquakes with location information of each submitted model. This permits graphical searching and downloading. Additionally, requests to view metadata for models launch the Metadata Browser and Editor (MOBE) for display of the metadata. Figures 1.14–1.16 illustrate the use of the RefEqViewer to search for models and browse the search results. The RefEqViewer will likely undergo further development to enhance the search panel and make it easier to identify the locations of models in the search results.

USGS
science for a changing world

Tools > WebForm

Webform Search Tool

Please use the form below to enter query information. Default values should return all data for a reference earthquake. Press *Search* when ready.

Collection:

Model Type:

Title:


Authors:

Keywords:
forward model
inverse model

Region:

	North	
	<input type="text" value="42"/>	
West		East
<input type="text" value="-125"/>		<input type="text" value="-114"/>
	South	
	<input type="text" value="32"/>	

Figure 1.12: The Webform search form for finding models using a web browser.



Tools > Webform > Search Results

Search Results

Your search returned the following model data and metadata. To revise your search, use the *back* button on your browser.

3 files found

Double difference relocated seismicity
metadata
Collection: PARKFIELD2004
File size: 5478400 bytes
Latitude / Longitude: 36.235486 (N), 35.688184 (S), -120.170833 (E), -120.817814 (W)
Authors: Jeanne Hardebeck
Version: 1.0.0
Version Description: Initial Submission

Double difference relocated seismicity
metadata
Collection: PARKFIELD2004
File size: 5478400 bytes
Latitude / Longitude: 36.235486 (N), 35.688184 (S), -120.170833 (E), -120.817814 (W)
Authors: Jeanne Hardebeck
Version: 1.0.1
Version Description: Fixed elevation metadata.

Relocated Seismicity Catalog
metadata
Collection: PARKFIELD2004
File size: 10240 bytes
Latitude / Longitude: 42 (N), 32 (S), -114 (E), -125 (W)
Authors: John Doe
Version: 1.0.0
Version Description: Initial Submission

Home
Tools
Documentation
About





Figure 1.13: Example of search results returned in a Webform search.

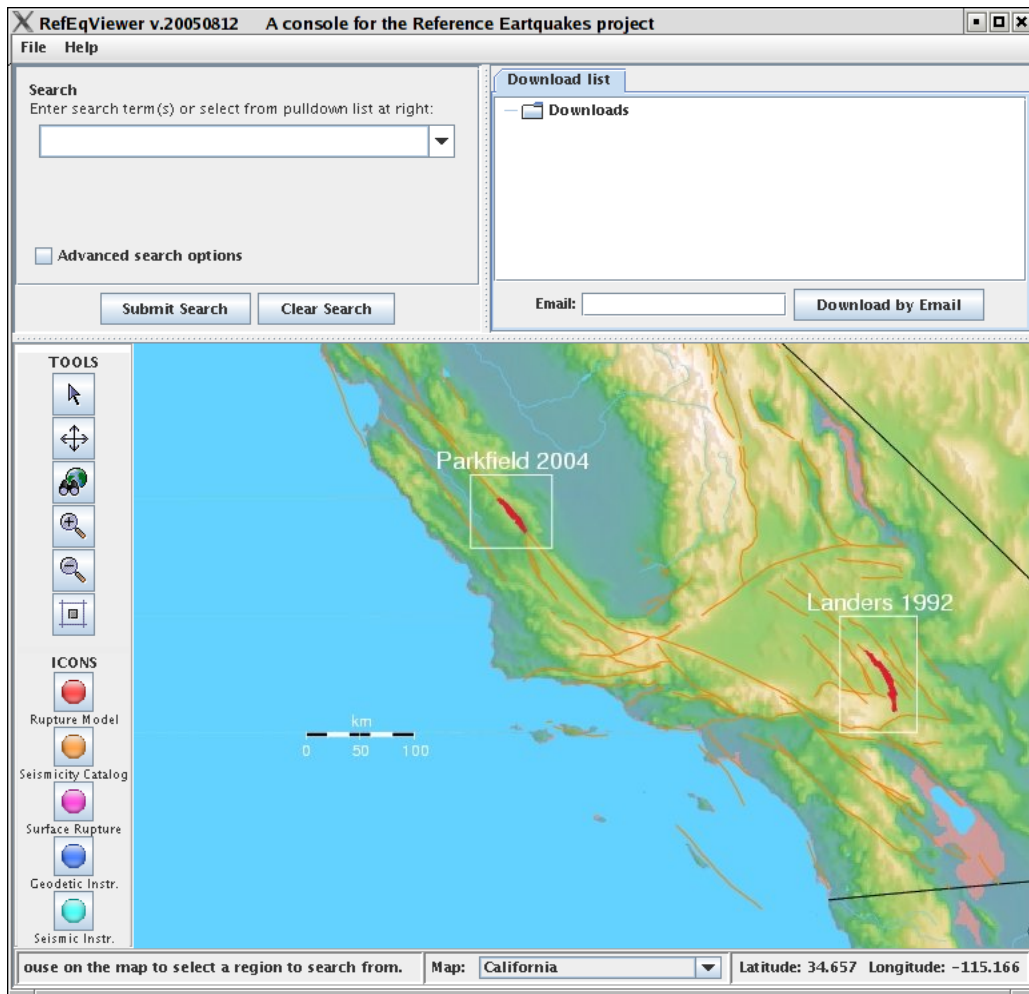


Figure 1.14: RefEqViewer upon startup. The window contains a map (lower portion of window), a search panel (upper left corner), and a results panel (upper right corner).

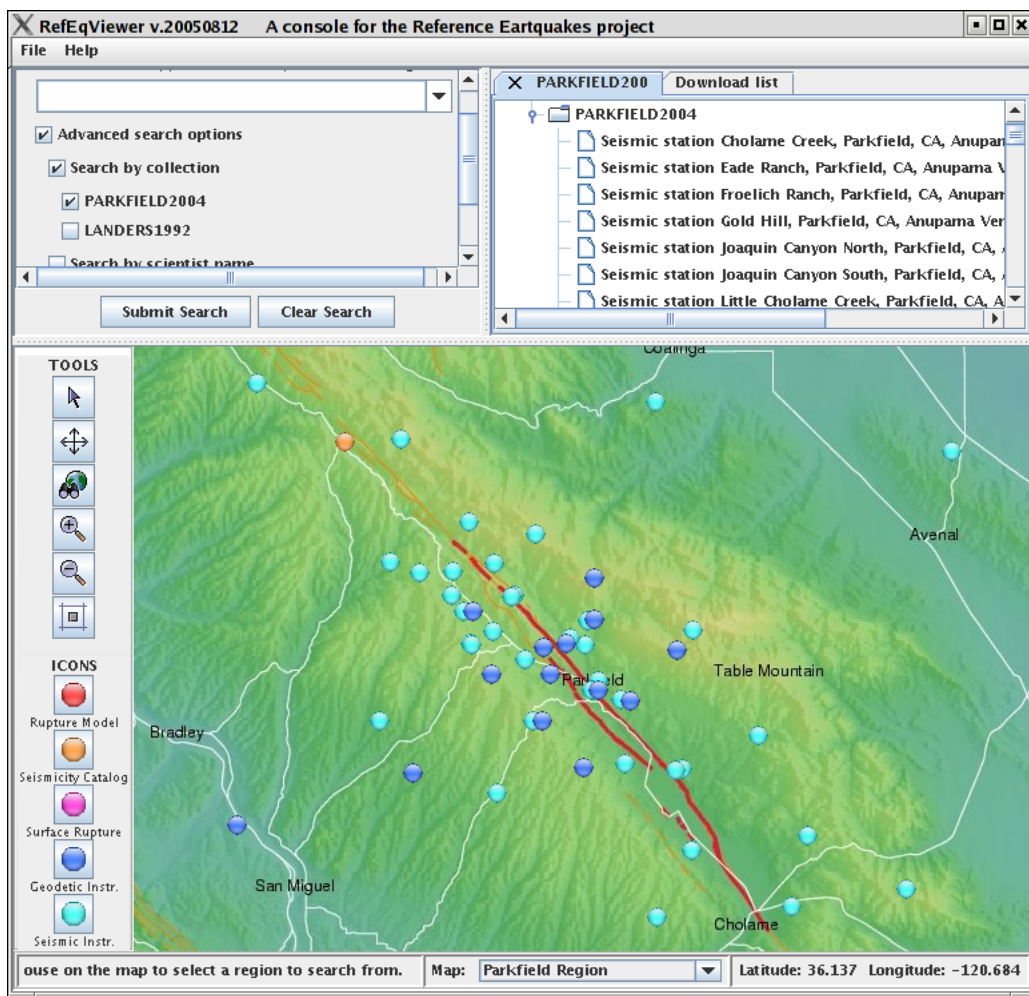


Figure 1.15: Browsing results in the RefEqViewer from a search of models related to the Parkfield reference earthquake. The center of each model is shown as a dot on the map and the results for each search are displayed in the upper right panel.

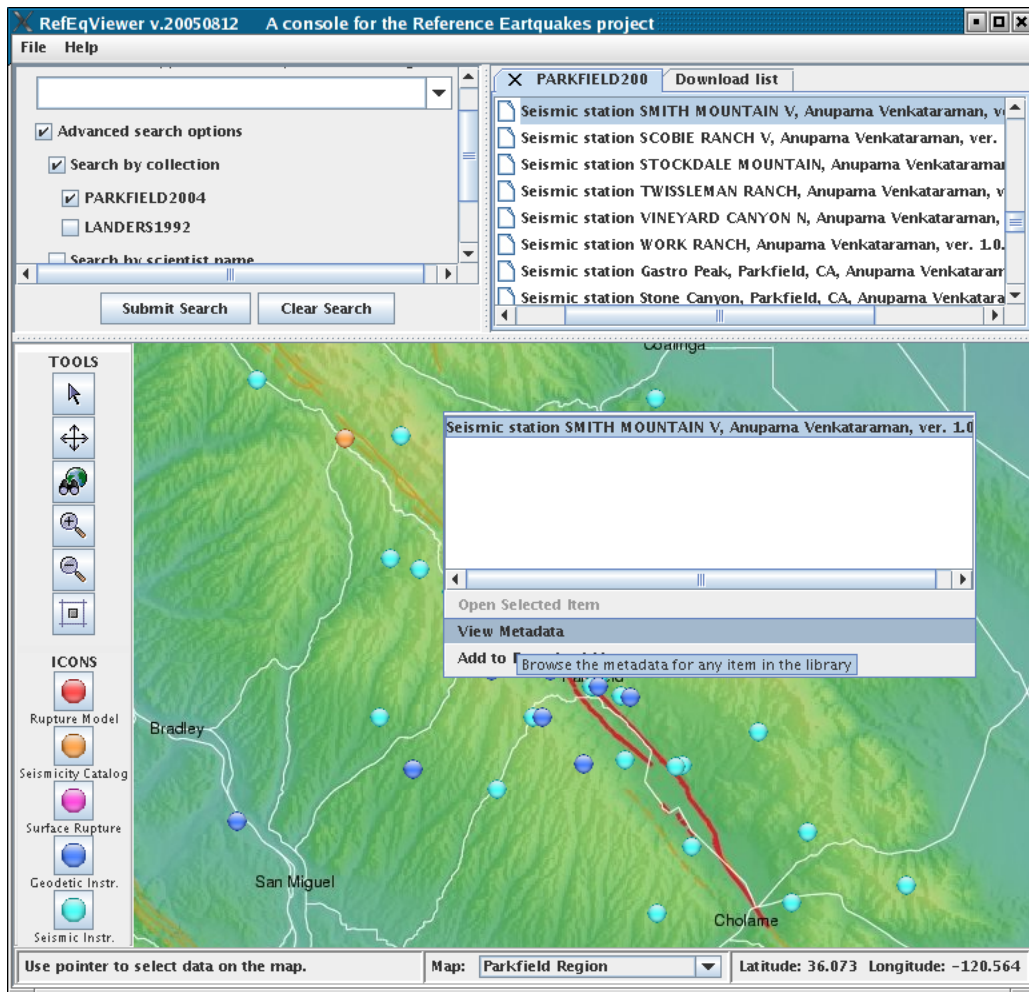


Figure 1.16: Browsing results in the RefEqViewer from a search of models related to the Parkfield reference earthquake. Clicking on the icons on the map displays a popup window that identifies the model and allows the user to launch the metadata viewer (MOBE) or add the model to the download list.

1.5 XML and HDF5

We chose to store the models using XML (eXtensible Markup Language) and HDF5 (Hierarchical Data Format) files to store the models in architecture independent formats, minimize the number of ways users must read/write model data, and make use of preexisting, widely available tools. Small models (seismicity catalogs, focal mechanism catalogs, GPS displacements, and surface rupture) are stored using XML files, whereas larger models (Coulomb stress change calculations, rupture models, and ground deformation) are stored using HDF5 files.

1.5.1 Overview of XML

XML has become a standard format for exchanging data among software components. The advantages it provides include the ability to make content flexible while adhering to a strict grammar, the files are self describing, and many tools exist to read/write XML files. The disadvantage is that few plotting packages can read XML files, so we have developed some utilities to help users extract data from XML files in a way that is compatible with popular plotting packages such as GMT and Matlab.

1.5.2 Overview of HDF5

HDF5 (Hierarchical Data Format; <http://hdf.ncsa.uiuc.edu/HDF5/>) is an efficient way to store large data files such as rupture models. Many tools exist for reading/writing HDF5 files from C/C++, Java, and Fortran 90 on most platforms, including massively parallel systems. The current release of Matlab (release 14) can read and write HDF5 files. We have created utilities and examples that demonstrate how to read/write model data from/to HDF5 files.

An HDF5 file is a container for storing a variety of scientific data and is composed of two primary types of objects: groups and datasets.

HDF5 group HDF5 groups are analogous to directories or folders of a file system. Groups can contain datasets, groups, and other objects.

HDF5 dataset Datasets are multidimensional arrays of data elements, together with supporting metadata.

HDF5 attribute Any HDF5 group or dataset may have an associated attribute list. An HDF5 attribute is a number or string that provides extra information about an HDF5 object.

Components of a *dataset* include:

1. Array: an ordered collection of identically typed data items distinguished by their indices (subscripts)
2. Dataspace: information about size and shape of a dataset array and selected parts of the array
3. User defined attribute list
4. Special storage options

Advantages of using HDF5 files:

- Each of the individual groups can be extracted independent of each other and thus ensures storage and data input/output efficiency.
- Large arrays can be stored efficiently in endian (byte order) transparent forms
- Support for key languages (FORTRAN, C, C++)

Disadvantages of using HDF5 files:

- More complex than writing ASCII or binary files.

2 Model Metadata

Metadata is simply defined as “structured data about data” and contains descriptive information about data. The metadata serves two purposes. First, it permits searching the library to find model and/or data terms associated with any of the metadata components. Second, it provides a mechanism for authors to annotate their files with information about the methods and data used to construct the models. The metadata for every entry in the digital library will include the Dublin Core components (standard components for digital libraries; <http://dublincore.org/documents/dcmi-terms/>) and additional components based on the model and observation types.

The metadata are harvested from the submission form. Some values correspond directly with form values while others are computed from the submission input (e.g., file size).

Table 2.1: Overview of metadata components

Description	Metadata Component
Basic (common to all)	Reference Earthquake
	Canonical Arbitrary Digital Object
Rupture Model	Rupture Model Core
	Geographic Extent
	Software
Seismicity Catalog	Seismicity Catalog Core
	Geographic Extent
	Software
Surface Rupture	Surface Rupture Core
	Geographic Extent

2.1 General Metadata Components

The general metadata components that are model-type independent are listed and detailed below. The reference earthquake, canonical, and geographic extent components are required for all model types. The software version components are required for models that are generated using computer software. The reference earthquake metadata is set by the digital library authors/maintainers when new earthquakes are added. Additionally, many of the canonical ADO components are set automatically.

Table 2.2: Reference Earthquake Metadata

Name of field	Brief Description
*Name	Name of earthquake (e.g., Landers and Parkfield)
*Geographic Region	Name of geographic region (e.g., Southern California)
*Origin Time	Date and time of earthquake (UTC; YYYY-MM-DD, HH:MM:SS.S)
*Moment Magnitude	Moment magnitude
*Harvard CMT Magnitude	Harvard CMT magnitude
*Hypocenter Latitude	Latitude of hypocenter (WGS84)
*Hypocenter Longitude	Longitude of hypocenter (WGS84)
*Hypocenter Depth	Depth of hypocenter (km)
*Latitude South	Approximate latitude of southern extent of rupture
*Latitude North	Approximate latitude of northern extent of rupture
*Longitude West	Approximate longitude of western extent of rupture
*Longitude East	Approximate longitude of eastern extent of rupture
*Mechanism	Normal/reverse/strike-slip, dip/rake/strike
#Faults	Name(s) of causative fault(s)
*Required	
#Optional	

Table 2.3: Canonical Arbitrary Data Object (ADO) Metadata

Name of field	Brief Description
*Title	Name of model
*Submitter	Name of person submitting the model
*Authors	Names of authors
*Affiliations	Author affiliations
*Keywords	Keywords to facilitate search
*Description	Brief description of data and method
*Date	Date submitted (YYYY-MM-DD)
*Type	Name of model type
*Format	Name of format (XML, HDF5)
*References	References to published papers, etc.
*Language	en (English)
*Relation	Relation to other model types (e.g., fault model used to calculate slip)
#Coverage	Geographic names associated with geographic extent
*Rights	Copyright details
#Parent	Relation to other model types (reserved for future use)
#Children	Relation to other model types (reserved for future use)
#Siblings	Relation to other model types (reserved for future use)
#Expert Level	Intended audience (undergraduate/graduate/researcher, etc.)
*Content Filenames	Filenames of files in model
*Filesize	File size
*Access Control	Access control (reserved for future use)
#Latitude North	Latitude of northern edge of bounding box (WGS84)
#Latitude South	Latitude of southern edge of bounding box (WGS84)
#Longitude West	Longitude of western edge of bounding box (WGS84)
#Longitude East	Longitude of eastern edge of bounding box (WGS84)
*Version Major	Model major version number
*Version Minor	Model minor version number
*Version Patch	Model version patch number
#Model Parent	Parent model (reserved for future use)
#Model Child	Child model (reserved for future use)
*Version Description	Description of model version
*Required	
#Optional	

Table 2.4: Geographic Extent Metadata

Name of field	Brief Description
*Latitude North	Latitude of northern edge of bounding box (WGS84)
*Latitude South	Latitude of southern edge of bounding box (WGS84)
*Longitude West	Longitude of western edge of bounding box (WGS84)
*Longitude East	Longitude of eastern edge of bounding box (WGS84)
*Horizontal Datum	Horizontal datum (WGS84)
*Elevation Minimum	Minimum elevation (km)
*Elevation Maximum	Maximum elevation (km)
*Vertical Datum	Vertical datum for elevation
#Geoid Model	Geoid used to calculate mean sea level
*Required	
#Optional	

Table 2.5: Software Metadata

Name of field	Brief Description
*Name	Name of software used to create model
*Version	Version number of software
*Description	Description of software
*Availability	Commercial/Academic/Private
#URL	URL if available
*Contact	Contact information
*Languages	Programming languages used in the software
*Required	
#Optional	

3 Rupture Model

Rupture models in the digital library describe slip on the fault(s) for the reference earthquakes. The rupture models can include coseismic and/or postseismic slip and may specify its temporal evolution. The rupture models may be forward or inverse models and may specify slip at vertices or on surface patches (subfaults).

3.1 Rupture Model Metadata

The complete set of metadata for rupture models includes the reference earthquake, canonical ADO, geographic extent, software, and rupture model specific metadata components. The values for these components are harvested from the submission form. The rupture model metadata components are shown in table 3.1.

Table 3.1: Rupture Model metadata

Name of field	Brief Description
*Forward Inverse	Forward or inverse model
*Time Dependence	Static, quasi-static, or dynamic model
*Parameters Description	Methodology/parameterization (e.g., multi-segment planar rectangles, rise time, stress drop, slip velocity function, constitutive law, initial stress conditions)
*Data Sources	Description of data sources
*Velocity Model Description	Description of velocity model
#Velocity Model Id	ID of seismic velocity model
*Begin Date Time	Begin date and time of rupture model
*Duration	Duration of model
*Frequency Min	Minimum frequency resolved in model
*Frequency Max	Maximum frequency resolved in model
*Data Filtering	Description of filters used on data
*Data Weighting	Description of weighting scheme
*Model Regularization	Description of model regularization
#Data Corrections	Description of data corrections
*Required	
#Optional	

3.2 Structure of HDF5 Files for Rupture Models

In this documentation, we describe in detail the HDF5 hierarchy that we use to store rupture models. To ensure consistency in the way rupture models are archived, we also define the terminology that we use.

In order to accommodate a wide variety of discretization methods for faults, we chose to store the fault topology and orientation information in addition to the slip values. The topology information stores the coordinates of the vertices and how the vertices are connected to form fault segments (please see definitions below). The orientation at a location is defined by the local strike, dip, and rake angles. In general, these angles define the nominal slip direction. Rake rotations are handled by specifying slip rate time histories with three components (slip associated with the rake angle, slip in the direction perpendicular to the rake angle, and fault opening). To allow quick access to basic information, we store both the final slip and the evolution of slip via snapshots of slip rate.

3.2.1 Definitions

Fault segment A simply connected surface making up part of a fault. Different segments of a fault can rupture in an earthquake, e.g., the Landers earthquake rupture is modeled as rupture on three major fault segments (Johnson, Homestead Valley and Emerson fault segments). Fault segments are composed of subfaults.

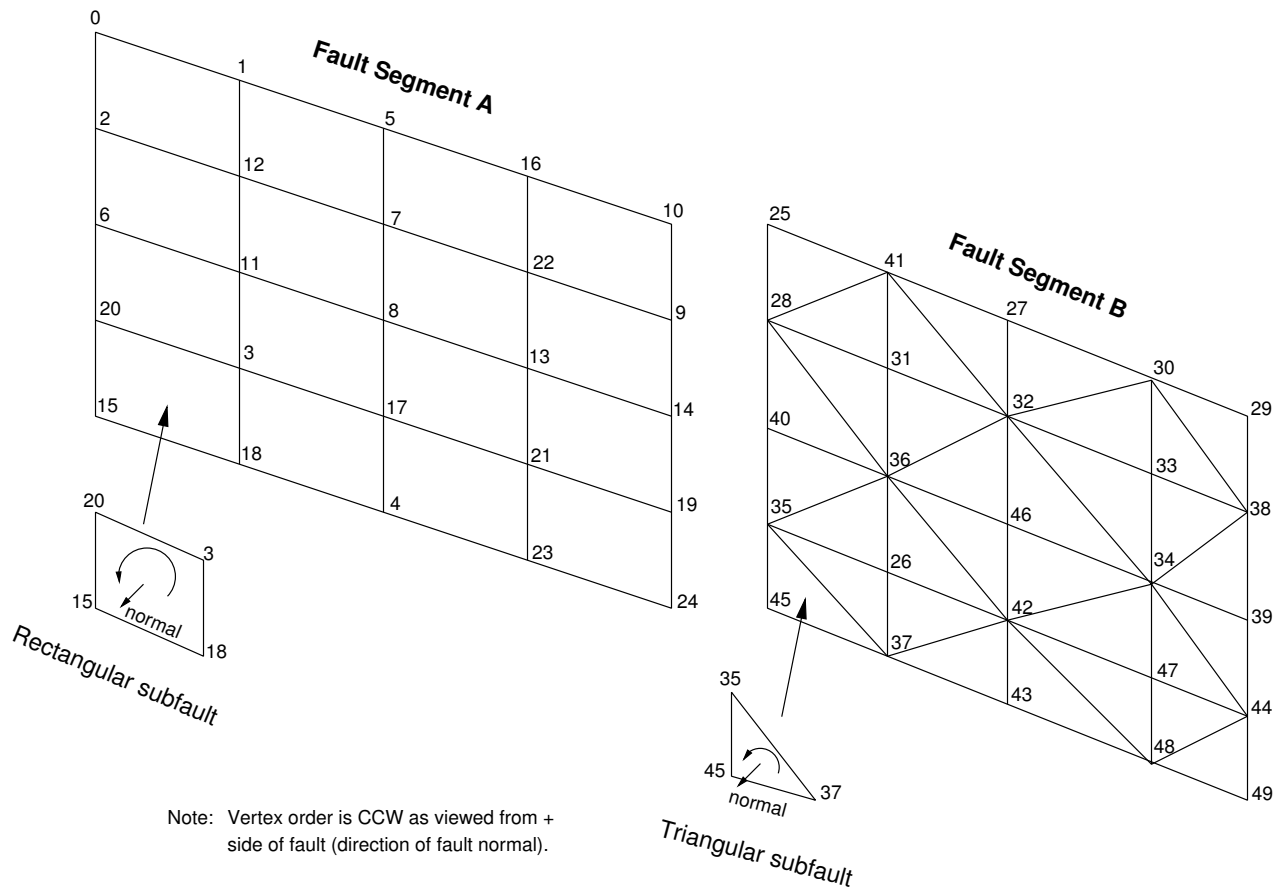


Figure 3.1: Two fault segments. For illustration purposes Fault Segment A is divided into square subfaults while Fault Segment B is divided into triangular subfaults. The vertices are numbered consecutively across all fault segments and the ordering for each subfault is counterclockwise as viewed from the positive side of the fault (direction to which fault normal points). Note that the direction of the ordering of vertices in each subfault must be consistent, but the vertices as a whole can be numbered in any order.

Subfault The smallest elements into which each fault segment is divided. These elements can be triangles or squares (as shown) or any other polygonal shape. The coordinates of the vertices define the geometry of the fault segment.

Vertex Each node point of a subfault (rectangular subfaults have 4 vertices and triangular subfaults have 3 vertices).

Vertex based rupture Rupture in which slip is defined at each vertex.

Subfault based rupture Rupture in which slip is defined on each subfault.

3.2.2 Conventions

- All arrays are 0 based, that is the index of the first entry is 0 (as opposed to 1).
- Vertex datasets are named ‘vertices’.
- Fault segment datasets are labeled with the name of the fault segment (e.g., ‘Homestead Valley’, ‘San Andreas’).
- The fault normal should be consistent across fault surfaces. The fault normal is defined by the cross product of the vector parallel to the rake direction and the vector perpendicular to the rake direction in the plane. The fault normal points from the footwall to the hanging wall (except when faults “roll over” and have dips greater than 90 degrees). This means that for a strike-slip fault, the rake-parallel slip is positive for left-lateral motion, the rake-perpendicular slip is positive for reverse motion, and the fault-opening displacement is positive for fault opening.
- Datasets may contain multidimensional arrays, but they are stored as one continuously indexed one-dimensional array. As a result, in multidimensional arrays it is important to know whether they are in column-major order or row-major order. In the specifications we explicitly show the order of the entries in multidimensional arrays to make the ordering clear.

3.2.2.1 Fault Orientation and Slip Vector

The orientation of the fault surface is defined by the strike and dip angles, ϕ and δ . The slip vector is defined by these two angles as well as the rake angle, λ . It is convenient to define axes rst as shown in figure 3.2 so that the r axis corresponds to the rake direction, the s axis corresponds to the in-plane rake perpendicular direction, and the t axis points from the footwall to the hanging wall. The rake perpendicular direction is chosen such that the rst axes are right-handed. The fault normal *must* be chosen so that it points in a consistent direction with respect to the surface. Choosing a consistent direction is particularly important when the dip of the fault varies about 90 degrees; with a consistent fault normal direction, the dip angle will be greater than 90 degrees when the fault “rolls over”. In general, this means that acceptable values for strike and dip range from 0 to 360 degrees.

This fault orientation convention leads to the convention that the slip components are positive for left-lateral, reverse, and fault opening dislocations. In rupture models without rake rotations it is convenient to reduce the number of nonzero slip components from 3 to 1. This is done by specifying the rake angle so that the rake direction is aligned with the slip vector. Alternatively, one could specify the rake angle as zero so that one in-plane slip component corresponds to lateral slip and the other corresponds to dip slip.

If we have a Cartesian coordinate system with $+x$ east, $+y$ north, and $+z$ up, then

$$\begin{pmatrix} r \\ s \\ t \end{pmatrix} = \begin{pmatrix} \sin \phi \cos \lambda - \cos \phi \cos \delta \sin \lambda & \cos \phi \cos \lambda + \sin \phi \cos \delta \sin \lambda & \sin \delta \sin \lambda \\ -\sin \phi \sin \lambda - \cos \phi \cos \delta \cos \lambda & -\cos \phi \sin \lambda + \sin \phi \cos \delta \cos \lambda & \sin \delta \cos \lambda \\ \cos \phi \sin \delta & -\sin \phi \sin \delta & \cos \delta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \quad (3.1)$$

In many instances it is useful to be able to compute the strike, dip, and rake angles given the rst axes. First, we compute the vector $\vec{r}_0 = (0, 0, 1) \times \vec{t}$, which is the r axis for a rake angle of zero. Using r_0 , we can find the strike

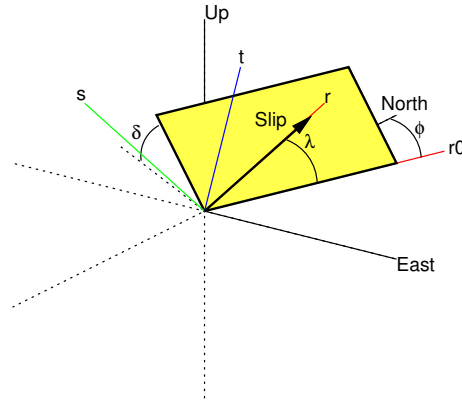


Figure 3.2: Fault orientation is defined by the strike and dip angles (ϕ and δ , respectively), and the slip vector is defined by the strike, dip, and rake angles (ϕ , δ , and λ , respectively).

and dip angles using

$$\phi = \arctan\left(\frac{r_{0x}}{r_{0y}}\right) \text{ where } \phi = \begin{cases} \phi + 2\pi & \text{if } \phi < 0 \text{ and } r_{0y} > 0 \\ \phi + \pi & \text{if } \phi < 0 \text{ and } r_{0y} < 0 \\ \phi + \pi & \text{if } \phi > 0 \text{ and } r_{0y} < 0 \end{cases} \quad (3.2)$$

and

$$\delta = \arctan\left(\frac{\sqrt{t_x^2 + t_y^2}}{t_z}\right) \text{ where } \delta = \begin{cases} \delta + \pi & \text{if } \delta < 0 \text{ and } s_{0z} > 0 \\ \pi - \delta & \text{if } \delta < 0 \text{ and } s_{0z} < 0 \\ 2\pi - \delta & \text{if } \delta > 0 \text{ and } s_{0z} < 0 \end{cases} \text{ and } s_{0z} = t_x r_{0y} - t_y r_{0x}. \quad (3.3)$$

The rake angle is the angle between \vec{r} and \vec{r}_0 , so

$$\lambda = \arcsin(|\vec{r}_0 \times \vec{r}|) \text{ where } \lambda = \pi - \lambda \text{ if } \vec{r}_0 \cdot \vec{r} < 0. \quad (3.4)$$

Note that ϕ and δ need to be adjusted because \arctan returns angles between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$, and λ needs to be adjusted because the magnitude of the cross product will always be positive.

3.3 Vertex based rupture models

Each of the items in figure 3.3 is explained below using examples to indicate the contents of the groups and datasets.

root (/) The root group contains only one group- the *rupture_model* group.

rupture_model This is the main group for rupture model information and it contains three groups - *topology* (information defining the fault surface), *slip_summary* (information about the slip orientation and final slip), and *slip_rate_th* (slip rate time history, information about the evolution of slip in time). The *topology* remains the same for the vertex based and subfault based rupture models, but the other two groups (*slip_summary* and *slip_rate_th*) will be different. Static models (ones without slip rate snapshots) do not contain the *slip_rate_th* group.

The attribute for this group is *slip_type*, which specifies whether the model is vertex based or subfault based (permissible values = {'vertices', 'subfaults'}).

topology This group contains (1) a dataset with the coordinates of all vertices comprising the fault surfaces and (2) a group of *fault_segments*.

vertices This dataset stores the coordinates of the vertices in terms of longitude, latitude, and elevation. The array size is number of vertices \times 3. This dataset might look something like:

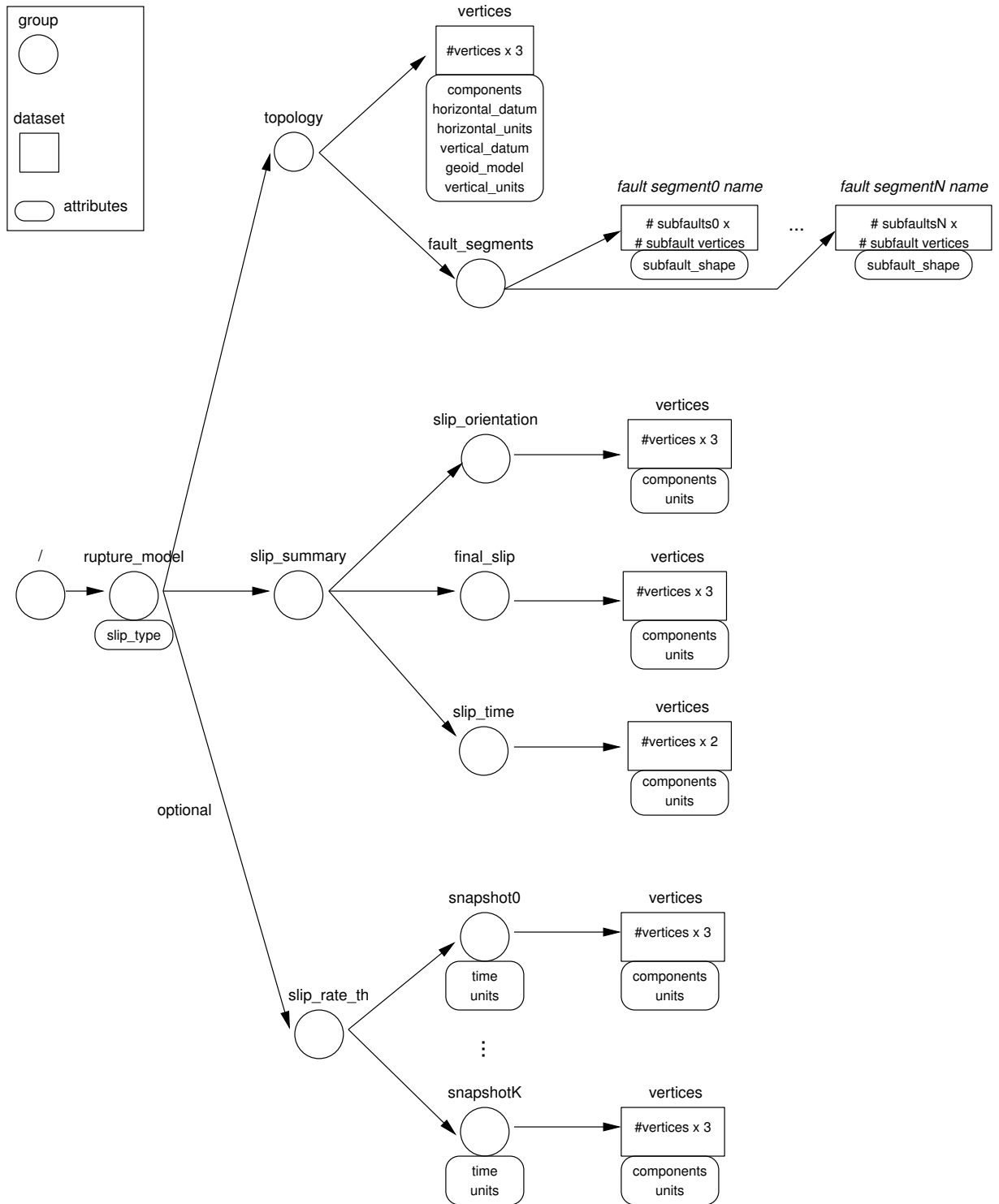


Figure 3.3: HDF5 hierarchy for vertex based rupture models.

Dataset: */rupture_model/topology/vertices*

	Lon (deg)	Lat (deg)	Elev (m)
Vertex 0	⁽⁰⁾ -120.500	⁽¹⁾ 36.000	⁽²⁾ 8.000
Vertex 1	⁽³⁾ -120.195	⁽⁴⁾ 36.015	⁽⁵⁾ 7.990
⋮	⋮	⋮	⋮
Vertex N	^(3N) -121.234	^(3N+1) 35.836	^(3N+2) 1.931

The attributes for this dataset are:

components Components of the coordinates (permissible value = {‘longitude, latitude, elevation’})

horizontal_datum Datum for longitude and latitude (permissible values = {‘WGS84’}),

horizontal_units Units for longitude and latitude (permissible values = {‘deg’}),

vertical_datum Datum for elevation (permissible values = {‘mean sea level’, ‘WGS84 ellipsoid’}),

geoid_model Geoid used to calculate mean sea level (useful if *vertical_datum* is ‘mean sea level’; permissible values = {‘mean sea level’, ‘WGS84 ellipsoid’}),

vertical_units Units for elevation {permissible values = ‘m’}).

fault_segments The *fault_segments* group is composed of a dataset for each simply connected fault surface (group size = number of fault segments). The datasets specify how the vertices are connected to form the geometric surface associated with the fault segment. The dataset array size is number of subfaults × number of subfault vertices. Each entry is the index of the vertex (first vertex has index 0). For example for ‘Fault Segment A’ in figure 3.1, the number of subfaults is 10 and the number of vertices per subfault is 4. The ordering of vertices in each subfault *does* matter. The convention is that the order is counter-clockwise (CCW) as viewed from the side of the fault to which the normal points (generally the hanging wall). This is consistent with the convention that the CCW orientation defines the normal direction using the right hand rule. For example, the *Fault Segment A* dataset would be:

Dataset: */rupture_model/topology/fault_segments/Fault Segment A*

	index 0	index 1	index 2	index 3
Subfault 0	⁽⁰⁾ 0	⁽¹⁾ 2	⁽²⁾ 12	⁽³⁾ 1
Subfault 1	⁽⁴⁾ 1	⁽⁵⁾ 12	⁽⁶⁾ 7	⁽⁷⁾ 5
Subfault 2	⁽⁸⁾ 5	⁽⁹⁾ 7	⁽¹⁰⁾ 22	⁽¹¹⁾ 16
Subfault 3	⁽¹²⁾ 16	⁽¹³⁾ 22	⁽¹⁴⁾ 9	⁽¹⁵⁾ 10
Subfault 4	⁽¹⁶⁾ 2	⁽¹⁷⁾ 6	⁽¹⁸⁾ 11	⁽¹⁹⁾ 12
⋮	⋮	⋮	⋮	⋮

slip_summary This group summarizes the slip at each vertex. The *slip_summary* contains three groups: *slip_orientation*, *final_slip* and *slip_time*. In the case of vertex based rupture models, these three groups each contain a single dataset, *vertices*.

slip_orientation This group stores the orientation of the slip for each vertex using the local strike, dip, and rake angles. The slip orientation dataset is constant in time, so rake rotations are implemented by using 3 components for slip and slip rate. The dataset array size is number of vertices × 3. Note that the fault normal must point in a consistent direction with respect to the fault surface, so strike, dip, and rake angles may each range between 0 and 360 degrees. The *vertices* dataset might be something like:

Dataset: */rupture_model/slip_summary/slip_orientation/vertices*

	Strike (deg)	Dip (deg)	Rake (deg)
Vertex 0	⁽⁰⁾ 57.0	⁽¹⁾ 85.6	⁽²⁾ 183.0
Vertex 1	⁽³⁾ 57.1	⁽⁴⁾ 85.8	⁽⁵⁾ 175.1
⋮	⋮	⋮	⋮
Vertex N	^(3N) 60.2	^(3N+1) 75.3	^(3N+2) 150.8

The attributes for this dataset are *components* to indicate the components in the dataset (permissible value = {'strike, dip, rake'}) and *units* to indicate the units for the strike, dip, and rake angles (permissible values = 'deg').

final_slip This group stores the final slip in the rupture model. The final slip is given with 3 components: slip parallel to the rake direction, slip perpendicular to the rake direction, and fault opening. The dataset array size is number of vertices \times 3. The *vertices* dataset might be something like:

Dataset: `/rupture_model/slip_summary/final_slip/vertices`

	Rake (m)	Rake \perp (m)	Opening (m)
Vertex 0	⁽⁰⁾ 0.24	⁽¹⁾ 0.03	⁽²⁾ 0.0
Vertex 1	⁽³⁾ 0.23	⁽⁴⁾ 0.06	⁽⁵⁾ 0.0
\vdots	\vdots	\vdots	\vdots
Vertex N	^(3N) 0.35	^(3N+1) -0.02	^(3N+2) 0.0001

The attributes for this dataset are *components*, which specifies the components of the dataset (permissible value = {'rake parallel, rake perpendicular, opening'}), and *units*, which specify the units for the slip (permissible values = 'm').

slip_time This dataset contains the time when slip begins and ends at each vertex relative to the origin time. If slip does not occur at a vertex or if the author believes slip is not resolved at a vertex, then the slip begin and end times should be set to 1.0e+30. For static models, the slip begin and end times should be set to 0.0. The dataset array size is number of vertices \times 2,

Dataset: `/rupture_model/slip_summary/slip_time/vertices`

	Slip begin time (sec)	Slip end time (sec)
Vertex 0	⁽⁰⁾ 0.44	⁽¹⁾ 1.27
Vertex 1	⁽²⁾ 0.45	⁽³⁾ 1.24
\vdots	\vdots	\vdots
Vertex N	^(2N) 8.35	^(2N+1) 10.23

The attributes for this dataset are *components*, which indicate the components of the dataset (permissible value = {'begin time, end time'}), and *units*, which indicate the units for the slip begin and end times (permissible values = 'sec').

slip_rate_th This group contains snapshots of the slip rate. This group only exists in dynamic models; static models should not include this group. Each snapshot of slip rate is stored in a different group. The number of groups = number of snapshots.

snapshotI This group contains the Ith snapshot of slip rate (I=[0..K] where K+1 is the number of snapshots). The slip rate is given in terms of the same 3 components as the final slip: slip rate parallel to the rake direction, slip rate perpendicular to the rake direction, and rate of fault opening. In vertex based rupture models, each snapshot contains a single dataset, *vertices*.

The attributes of this group are *time*, which specifies the time of slip rate snapshot as a single precision floating point number and *units*, which specifies the units for time of snapshot (permissible values = 'sec'). The dataset array size is number of vertices \times 3. The *vertices* dataset might be something like:

Dataset: `/rupture_model/slip_rate_th/snapshot0/vertices`

	Rake (m/s)	Rake \perp (m/s)	Opening (m/s)
Vertex 0	⁽⁰⁾ 0.78	⁽¹⁾ 0.04	⁽²⁾ 0.0
Vertex 1	⁽³⁾ 0.96	⁽⁴⁾ 0.18	⁽⁵⁾ 0.0
\vdots	\vdots	\vdots	\vdots
Vertex N	^(3N) 1.34	^(3N+1) -0.14	^(3N+2) 0.001

The attributes for this dataset are *components*, which specifies the components of the slip rate dataset (permissible value = {'rake parallel, rake perpendicular, opening'}), and *units*, which specifies the units for slip rate (permissible values = 'm/s').

3.4 Subfault based rupture models

Each of the items in figure 3.4 is explained below using examples to indicate the contents of the groups and datasets.

root (/) The root group contains only one group- the *rupture_model* group.

rupture_model This is the main group for rupture model information and it contains three groups - *topology* (information defining the fault surface), *slip_summary* (information about the slip orientation and final slip), and *slip_rate_th* (slip rate time history, information about the evolution of slip in time). The *topology* remains the same for the vertex based and subfault based rupture models, but the other two groups (*slip_summary* and *slip_rate_th*) will be different. Static models (ones without slip rate snapshots) do not contain the *slip_rate_th* group.

The attribute for this group is *slip_type*, which specifies whether the model is vertex based or subfault based (permissible values = {'vertices', 'subfaults'}).

topology This group contains (1) a dataset with the coordinates of all vertices comprising the fault surfaces and (2) a group of *fault_segments*.

vertices This dataset stores the coordinates of the vertices in terms of longitude, latitude, and elevation. The array size is number of vertices \times 3. This dataset might look something like:

Dataset: */rupture_model/topology/vertices*

	Lon (deg)	Lat (deg)	Elev (m)
Vertex 0	⁽⁰⁾ -120.500	⁽¹⁾ 36.000	⁽²⁾ 8.000
Vertex 1	⁽³⁾ -120.195	⁽⁴⁾ 36.015	⁽⁵⁾ 7.990
⋮	⋮	⋮	⋮
Vertex N	^(3N) -121.234	^(3N+1) 35.836	^(3N+2) 1.931

The attributes for this dataset are:

components Components of the coordinates (permissible value = {'longitude, latitude, elevation'})

horizontal_datum Datum for longitude and latitude (permissible values = {'WGS84'}),

horizontal_units Units for longitude and latitude (permissible values = {'deg'}),

vertical_datum Datum for elevation (permissible values = {'mean sea level', 'WGS84 ellipsoid'}),

geoid_model Geoid used to calculate mean sea level (useful if *vertical_datum* is 'mean sea level'), (permissible values = {'mean sea level', 'WGS84 ellipsoid'})

vertical_units Units for elevation {permissible values = {'m'}}.

fault_segments The *fault_segments* group is composed of a dataset for each simply connected fault surface (group size = number of fault segments). The datasets specify how the vertices are connected to form the geometric surface associated with the fault segment. The dataset array size is number of subfaults \times number of subfault vertices. Each entry is the index of the vertex (first vertex has index 0). For example for 'Fault Segment A' in figure 3.1, the number of subfaults is 10 and the number of vertices per subfault is 4. The ordering of vertices in each subfault *does* matter. The convention is that the order is counter-clockwise (CCW) as viewed from the side of the fault to which the normal points (generally the hanging wall). This is consistent with the convention that the CCW orientation defines the normal direction using the right hand rule. For example, the *Fault Segment A* dataset would be:

Dataset: */rupture_model/topology/fault_segments/Fault Segment A*

	index 0	index 1	index 2	index 3
Subfault 0	⁽⁰⁾ 0	⁽¹⁾ 2	⁽²⁾ 12	⁽³⁾ 1
Subfault 1	⁽⁴⁾ 1	⁽⁵⁾ 12	⁽⁶⁾ 7	⁽⁷⁾ 5
Subfault 2	⁽⁸⁾ 5	⁽⁹⁾ 7	⁽¹⁰⁾ 22	⁽¹¹⁾ 16
Subfault 3	⁽¹²⁾ 16	⁽¹³⁾ 22	⁽¹⁴⁾ 9	⁽¹⁵⁾ 10
Subfault 4	⁽¹⁶⁾ 2	⁽¹⁷⁾ 6	⁽¹⁸⁾ 11	⁽¹⁹⁾ 12
⋮	⋮	⋮	⋮	⋮

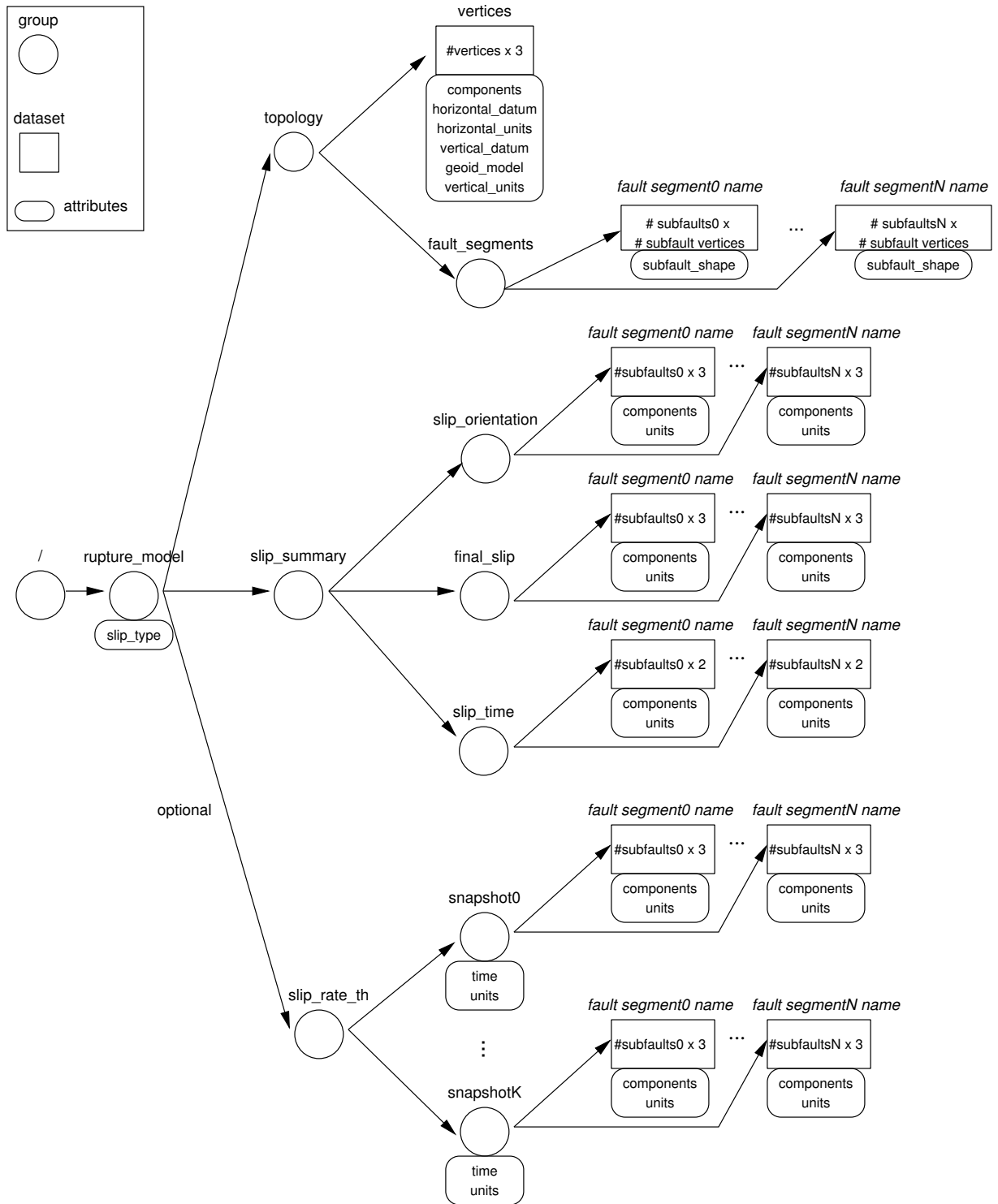


Figure 3.4: HDF5 hierarchy for subfault based rupture models.

slip_summary This group summarizes the slip on each subfault. The *slip_summary* contains three groups: *slip_orientation*, *final_slip* and *slip_time*. In subfault based rupture models, these three groups each contain a dataset for each fault segment, where the dataset name matches the name of the fault segment used in the *topology* group.

slip_orientation This group stores the orientation of the slip for each subfault using the local strike, dip, and rake angles. The slip orientation dataset is constant in time, so rake rotations are implemented by using 3 components for slip and slip rate. The dataset array size is number of subfaults \times 3. Note that the fault normal must point in a consistent direction with respect to the fault surface, so strike, dip, and rake angles may each range between 0 and 360 degrees. The dataset for each fault segment might be something like:

Dataset: */rupture_model/slip_summary/slip_orientation/Fault Segment A*

	Strike (deg)	Dip (deg)	Rake (deg)
Subfault 0	⁽⁰⁾ 57.0	⁽¹⁾ 85.6	⁽²⁾ 183.0
Subfault 1	⁽³⁾ 57.1	⁽⁴⁾ 85.8	⁽⁵⁾ 175.1
⋮	⋮	⋮	⋮
Subfault M	^(3M) 60.2	^(3M+1) 75.3	^(3M+2) 150.8

The attributes for this dataset are *components* to indicate the components in the dataset (permissible value = {‘strike, dip, rake’}) and *units* to indicate the units for the strike, dip, and rake angles (permissible values = ‘deg’).

final_slip This group stores the final slip in the rupture model. The final slip is given with 3 components: slip parallel to the rake direction, slip perpendicular to the rake direction, and fault opening. The dataset array size is number of subfaults \times 3. The dataset for each fault might be something like:

Dataset: */rupture_model/slip_summary/final_slip/Fault Segment A*

	Rake (m)	Rake \perp (m)	Opening (m)
Subfault 0	⁽⁰⁾ 0.24	⁽¹⁾ 0.03	⁽²⁾ 0.0
Subfault 1	⁽³⁾ 0.23	⁽⁴⁾ 0.06	⁽⁵⁾ 0.0
⋮	⋮	⋮	⋮
Subfault M	^(3M) 0.35	^(3M+1) -0.02	^(3M+2) 0.01

The attributes for this dataset are *components*, which specifies the components of the dataset (permissible value = {‘rake parallel, rake perpendicular, opening’}), and *units*, which specify the units for the slip (permissible values = ‘m’).

slip_time This dataset contains the time when slip begins and ends on each subfault relative to the origin time. If slip does not occur on a subfault or if the author believes slip is not resolved on a subfault, then the slip begin and end times should be set to 1.0e+30. For static models, the slip begin and end times should be set to 0.0. The dataset array size is number of subfaults \times 2. The dataset for each fault might be something like:

Dataset: */rupture_model/slip_summary/slip_time/Fault Segment A*

	Slip begin time (sec)	Slip end time (sec)
Subfault 0	⁽⁰⁾ 0.44	⁽¹⁾ 1.27
Subfault 1	⁽²⁾ 0.45	⁽³⁾ 1.24
⋮	⋮	⋮
Subfault M	^(2M) 8.35	^(2M+1) 10.23

The attributes for this dataset are *components*, which indicate the components of the dataset (permissible value = {‘begin time, end time’}), and *units*, which indicate the units for the slip begin and end times (permissible values = ‘sec’).

slip_rate_th This group contains snapshots of the slip rate. This group only exists in dynamic models; static models should not include this group. Each snapshot of slip rate is stored in a different group. The number of groups = number of snapshots.

snapshot This group contains the I th snapshot of slip rate ($I=[0..K]$ where $K+1$ is the number of snapshots). The slip rate is given in terms of the same 3 components as the final slip: slip rate parallel to the rake direction, slip rate perpendicular to the rake direction, and rate of fault opening. In subfault based rupture models, each snapshot contains a dataset for each fault segment, where the dataset name matches the name of the fault segment used in the *topology* group.

The attributes of this group are *time*, which specifies the time of slip rate snapshot as a single precision floating point number and *units*, which specifies the units for time of snapshot (permissible values = 'sec'). The dataset array size is number of subfaults \times 3. The dataset for each fault might be something like:

Dataset: */rupture_model/slip_rate.th/snapshot0/Fault Segment A*

	Rake (m/s)	Rake \perp (m/s)	Opening (m/s)
Subfault 0	⁽⁰⁾ 0.78	⁽¹⁾ 0.04	⁽²⁾ 0.0
Subfault 1	⁽³⁾ 0.96	⁽⁴⁾ 0.18	⁽⁵⁾ 0.0
\vdots	\vdots	\vdots	\vdots
Subfault M	^(3M) 1.34	^(3M+1) -0.14	^(3M+2) 0.001

The attributes for this dataset are *components*, which specifies the components of the slip rate dataset (permissible value = {'rake parallel, rake perpendicular, opening'}), and *units*, which specifies the units for slip rate (permissible values = 'm/s').

3.4.1 Utilities for Reading/Writing Rupture Models

3.4.1.1 HDF5 Utilities

HDF5 contains utilities to enable users to examine the contents of HDF5 files. `h5dump` will dump the contents of HDF5 files to an ASCII file or stdout. The amount and type of information dumped is controlled by command line arguments. Similarly, `h5ls` will list information about selected parts of an HDF5 file. For more information on how to use these programs, see the HDF5 documentation or run them with the `-h` flag.

3.4.1.2 Reading/Writing using C

We have written a C library to make it easier to read/write information from/to rupture models in HDF5 files. The library insulates the user from HDF5 routines by consolidating HDF5 operations. The library is distributed with examples for writing both vertex based and subfault based rupture models and reading datasets and attributes. A tarball containing the source code for the library can be downloaded from the documentation section of the digital library web site, <http://www.wr.usgs.gov>.

3.4.1.3 Reading/Writing using Matlab

Matlab version 7.0.0 (release 14) and later include functions for reading/writing HDF5 files. `hdf5read` and `hdf5write` read and write datasets and attributes, respectively. `hdf5info` can be used to extract information about the contents of the HDF5 file.

4 Seismicity Catalog

The digital library stores seismicity catalogs associated with the reference earthquake. In general, these are events that have been relocated using a local velocity model, double difference travel times, and/or other algorithms to give greater location accuracy than the original data center catalog locations. The catalogs may also provide locations for events not in the data center catalogs. Each seismicity catalog is stored as an XML file accompanied by a metadata file.

4.1 Seismicity Catalog Metadata

The complete set of metadata for the seismicity catalog includes the reference earthquake, canonical ADO, geographic extent, software, and seismicity catalog specific metadata components. The values for these components are harvested from the submission form. The seismicity catalog metadata components are shown in table 4.1.

Table 4.1: Seismicity Catalog Metadata

Name of field	Brief Description
*Number of Events	Number of events in the catalog
*Begin Date	Begin date of catalog (YYYY-MM-DD)
*End Date	End date of catalog (YYYY-MM-DD)
*Magnitude Min	Min magnitude in catalog
*Magnitude Max	Max magnitude in catalog
*Velocity Model Description	Description of velocity model
#Velocity Model Id	ID of seismic velocity model
*Required	
#Optional	

4.2 Structure of XML Files for the Seismicity Catalog

In the following section we describe in detail the elements in the XML file used to store a seismicity catalog.

header The header follows the same format as the header of an XHTML file. You can include any information about a seismicity catalog you wish in the header by repeating *meta* elements with attributes *name* and *content*, for example

```
<meta name="number of events" content="1" />
```

events This is the container element for all of the events in the seismicity catalog.

event This element is the container for each event in the catalog. It is repeated for each event.

identification This element contains the elements *eventid* and *clusterid*.

eventid A string that gives the unique identity of each event. This is a required field.

clusterid When events are part of a cluster of events, especially during relative relocation of a group of events, it is useful to know the cluster id. This is an optional field.

location A required set of elements, event location has the child elements *latitude*, *longitude* and *elevation*.

latitude Event latitude in degrees in WGS84

longitude Event longitude in degrees in WGS84

elevation Event elevation in meters (below sea level or the reference ellipsoid is negative)

The attributes for *location* are:

- horizontal_datum*** Datum for longitude and latitude (permissible values = {‘WGS84’}),
- units*** Units for longitude and latitude (permissible values = {‘deg’}),
- vertical_datum*** Datum for elevation (permissible values = {‘mean sea level’, ‘WGS84 ellipsoid’}),
- geoid_model*** Geoid used to calculate mean sea level (needed only if *vertical_datum* is ‘mean sea level’; permissible values = {‘mean sea level’, ‘WGS84 ellipsoid’}),
- units*** Units for elevation {permissible values = ‘m’}).
- origin_time*** A required element that stores the event origin time in the dateTime format (e.g., YYYY-MM-DDThh:mm:ss.SZ for UTC)
- magnitude*** A required element that stores the event magnitude information.
The attributes for *magnitude* are *type* for magnitude type (e.g., moment magnitude) and *source* for the source from which the magnitude value is derived (e.g., SCEDC)
- processing*** An optional field that stores information about the data used to produce the seismicity catalog.
- p_picks*** The number of P wave picks used for determining event locations.
- s_picks*** The number of S wave picks used for determining event locations.
- uncertainty*** An optional element that stores information on measures of location uncertainty.
- relative_relocation*** Uncertainties associated with relative relocation. This is an optional element.
- horizontal*** The location uncertainty in the horizontal direction.
The attributes for horizontal uncertainty are *units* for horizontal uncertainty {permissible values = ‘m’}).
- vertical*** The location uncertainty in the vertical direction.
The attributes for vertical uncertainty are *units* for vertical uncertainty {permissible values = ‘m’}).
- absolute_relocation*** Uncertainties associated with absolute relocation. This is an optional element.
- horizontal*** The location uncertainty in the horizontal direction.
The attributes for horizontal uncertainty are *units* for horizontal uncertainty {permissible values = ‘m’}).
- vertical*** The location uncertainty in the vertical direction.
The attributes for vertical uncertainty are *units* for vertical uncertainty {permissible values = ‘m’}).
- rms_travel_time_residual*** Uncertainty can also be given by the root mean squared value of the travel time residual.
The attributes for *rms_travel_time_residual* are *units* for the rms travel time residual {permissible values = ‘sec’}).

4.2.1 Example of an XML file for seismicity catalog

An example file showing the structure of the XML file for a seismicity catalog is shown in figure 4.1.

4.2.2 Utilities for Reading/Writing Seismicity Catalog Files

Software producing relocated seismicity catalogs, such as HypoDD, as well as software for plotting event locations, such as GMT and Matlab, use simple column formats with one event per line. To facilitate conversion between these simple formats and the seismicity catalog XML files, we have created scripts to convert column formats to/from XML. These scripts are written in Python but are setup so that users do not need to know Python. The scripts are available from the documentation section of the Reference Earthquakes Digital Library web site (<http://www.wr.usgs.gov>).

```

<?xml version="1.0" encoding="utf-8"?>
<seismicity_catalog
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="seismicity_catalog.xsd">
  <header>
    <meta name="number of events" content="1"/>
  </header>
  <events>
    <event>
      <identification>
        <eventid source="SCEDC">xx00000</eventid>
        <clusterid>-1</clusterid>
      </identification>
      <location horizontal_datum="WGS84" vertical_datum="mean sea level"
        geoid="GEOID93">
        <longitude units="deg">-120.355</longitude>
        <latitude units="deg">35.805</latitude>
        <elevation units="m">-10000.0</elevation>
      </location>
      <origin_time>2005-05-26T06:32:48.3Z</origin_time>
      <magnitude type="see data center" source="data center">3.4</magnitude>
      <processing>
        <p_picks>32</p_picks>
        <s_picks>26</s_picks>
      </processing>
      <uncertainty>
        <relative_location>
          <horizontal units="m">200</horizontal>
          <vertical units="m">800</vertical>
        </relative_location>
        <rms_travel_time_residual units="sec">0.001</rms_travel_time_residual>
      </uncertainty>
    </event>
  </events>
</seismicity_catalog>

```

Figure 4.1: Sample XML file for a seismicity catalog. The catalog contains only one event. Note that the sample XML file contains several optional elements, such as *clusterid*, *relative_location*, and *rms_travel_time_residual*, but omits the optional element *absolute_location*. Bold indicates element values, green, attribute names, and red, attribute values.

4.2.2.1 Converting to XML

The script `seiscatalog_hypodd2xml.py` will convert HypoDD output to a seismicity catalog XML file. The script uses command line arguments to set the names of the input and output files. Variations from the HypoDD format are accommodated with optional command line arguments and editing the script to change how columns are parsed. Optional command line arguments include specification of the column delimiter, a flag for comments, and the number of columns in the input file. More information about the command line arguments can be found by running the script with the `-h` flag. In order to change how the columns are parsed, you edit the function `parseEvent` defined at the top of the script (see the script for further details).

4.2.2.2 Converting from XML

The script `seiscatalog_xml2txt.py` will extract information from a seismicity catalog XML file and write it out in columns. The script uses command line arguments to specify the names of the input and output files (run the script with the `-h` flag for more information). The functions `writeHeader` and `writeEvent` define the output header information and the format of each event. More detailed information about setting up these functions can be found in the script.

5 Surface Rupture

The digital library contains observations of surface rupture for the reference earthquakes. These may be related to observations of surface cracking (e.g., en echelon cracks along a fault trace) or of slip along the fault trace measured, for example, using alinement arrays.

5.1 Surface Rupture Metadata

A surface rupture model consists of the estimated strike-slip, dip-slip and / or opening for different points along a surface rupture. This is derived from measurements of the orientation and amount of offset observed.

The complete set of metadata for surface rupture models includes the reference earthquake, canonical ADO, geographic extent, and surface rupture model specific metadata components. The values for these components are harvested from the submission form. The surface rupture model metadata components are shown in table 5.1.

Table 5.1: Surface Rupture Model metadata

Name of field	Brief Description
*Survey Begin Date	Begin date of survey (YYYY-MM-DD)
*Survey End Date	End date of survey (YYYY-MM-DD)
*Measurement Type	Type of measurement, e.g. mapping, alinement array
*Number of measurements	Number of measurements in the data file
*Required	
#Optional	

5.2 Structure of XML Files for Surface Rupture Models

The surface rupture observations are stored in XML files. The format of these files is defined by a schema. The schema dictates the acceptable number and form of each piece of surface rupture information, including which items are required.

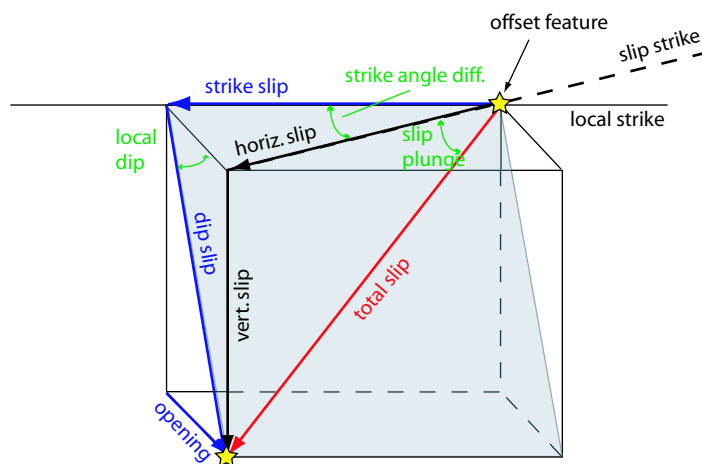


Figure 5.1: Diagram of a surface rupture feature and the quantities that may be measured in the field. The names used here correspond to those used in the XML file.

For surface rupture observations based on mapping, the information that will be stored is based on the diagram in figure 5.1. Surface rupture models based on alinement array or other measurement types have not yet been implemented.

The following is a detailed description of each piece of information. Following that, in figure 5.2 is an example of the XML file.

header The header follows the same format as the header of an XHTML file. You can include any information about a surface rupture model you wish in the header by repeating *meta* elements with attributes *name* and *content*, for example

```
<meta name="number of events" content="1" />
```

measurements This is the main element, and it contains information collected at various locations. Each location's data is stored in a separate *measurement* element. A minimum of one *measurements* element is required.

measurement Each of these contains information for data collected at a specific location. Each *measurement* element contains the following elements:

observer This element must occur once and contains the first and last name of the person who made the measurement stored in the *first_name* and *last_name* elements.

first_name Observer's first name; this element must occur once.

last_name Observer's last name; this element must occur once.

location This element must occur once and contains the location where the measurement was made, stored in the following elements.

longitude This element must occur once and should contain one value.

This element has the required attribute *units* which is fixed to 'deg'.

latitude This element must occur once and should contain one value.

This element has the required attribute *units* which is fixed to 'deg'.

elevation This element is optional; if used, one value should be input.

This element has the required attribute *units* which is fixed to 'm'.

The *location* element has the following attributes:

horizontal_datum Horizontal datum for longitude and latitude (permissible values = {'WGS84'}).

This attribute is required.

vertical_datum Vertical datum for elevation (permissible values = {'WGS84 ellipsoid', 'mean sea level'}). This attribute is required.

geoid_model Optional attribute used to specify the geoid used to calculate mean sea level. Provide this attribute if the *vertical_datum* is mean sea level.

obs_time The date and time when the observation was made. This element must appear once. Times should be given in UTC time. The format is YYYY-MM-DDThh:mm:ssZ. YYYY is the 4-digit year, MM is the month, and DD is the day. hh is the hour, mm is the minutes, and ss is seconds. The seconds may have decimal places. Note the T separating the date and time, and the final Z (which indicates UTC time).

photo_link This element can be used to store URL(s) to photographs of the surface rupture at this location. This element may appear zero or one times and contains the following element

src Contains the URL. This element may be repeated as necessary to store multiple links (one link per element).

fault This element contains information about the fault being observed. It is optional.

name This element contains the name of the fault being observed. It is optional.

local_strike This element stores the local strike of the fault at this location measured clockwise from north; it is optional. This differs from the azimuth of an observed surface offset feature, which is discussed below.

If used, this element has the required attribute *units* which is fixed to 'deg'.

local_dip This element stores the local dip of the fault at this location measured 90 degrees clockwise from strike; it is optional.

If used, this element has the required attribute *units* which is fixed to 'deg'.

- cracks_observed*** This element contains a boolean value. These values can be true or 1 (1 indicates true), false or 0 (0 indicates false). The value indicates whether any surface rupture was observed at this location. This is a required element. If surface rupture was observed, information regarding the rupture is stored in the *slip_data* element.
- slip_data*** This element is used to store the raw measurements and the quantities derived from these measurements for this location. A minimum of one *slip_data* element is required if *cracks_observed* is true. More *slip_data* elements can be included if multiple measurements were made at the same location. If *cracks_observed* is false, there are no *slip_data* elements. The observed and derived quantities are stored in the following elements.
- slip_strike*** This element stores the strike of the observed surface rupture feature (e.g. crack) at this location. This element appears 0 or 1 times, depending on whether the strike was measured here.
This element has the required attribute *units* which is fixed to ‘deg’.
- strike_angle_diff*** This element stores the acute angle between the local strike of the fault and the strike of the observed surface rupture feature at this location. This element appears 0 or 1 times, depending on whether the angle difference was calculated for this location.
This element has the required attribute *units* which is fixed to ‘deg’.
- slip_plunge*** This element stores the plunge of the observed offset feature at this location. This element appears 0 or 1 times, depending on whether the plunge was measured here.
This element has the required attribute *units* which is fixed to ‘deg’.
- total_slip*** This element stores the total offset across the observed surface rupture feature (e.g. crack) at this location. This element appears 0 or 1 times, depending on whether the offset was measured here.
This element has the required attribute *units* which is fixed to ‘m’.
- horiz_slip*** This element stores the horizontal offset across the observed surface rupture feature (e.g. crack) at this location. This element appears 0 or 1 times, depending on whether the horizontal offset was measured here.
This element has the required attribute *units* which is fixed to ‘m’.
- vert_slip*** This element stores the vertical offset across the observed surface rupture feature (e.g. crack) at this location. This element appears 0 or 1 times, depending on whether the vertical offset was measured here.
This element has the required attribute *units* which is fixed to ‘m’.
- strike_slip*** This element stores the strike slip across the observed surface rupture feature (e.g. crack) at this location. This value is derived from the horizontal offset and the difference between the strike of the surface rupture and the local strike of the fault. This element appears 0 or 1 times, depending on whether it was calculated for this location.
This element has the required attribute *units*, which is fixed to ‘m’, and the optional attribute *uncertainty* in which is stored the uncertainty associated with the calculated strike slip.
- dip_slip*** This element stores the dip slip across the observed surface rupture feature (e.g. crack) at this location. This value is derived from the vertical offset and the local dip. This element appears 0 or 1 times, depending on whether it was calculated for this location.
This element has the required attribute *units*, which is fixed to ‘m’, and the optional attribute *uncertainty* in which is stored the uncertainty associated with the calculated dip slip.
- opening*** This element stores the opening across the observed surface rupture feature (e.g. crack) at this location. This element appears 0 or 1 times, depending on whether it was measured for this location.
This element has the required attribute *units*, which is fixed to ‘m’, and the optional attribute *uncertainty* in which is stored the uncertainty associated with the calculated opening.

5.2.1 Utilities for Reading/Writing Surface Rupture Files

We have developed utilities for generating surface rupture XML files and for extracting information from these XML files. These utilities are Python scripts but are setup so that users do not need to know Python.

```

<?xml version="1.0" encoding="utf-8"?>
<surface_rupture
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="surface_rupture.xsd">
  <header>
    <meta name="number of measurements" content="1"/>
  </header>
  <measurements>
    <measurement>
      <observer>
        <first_name>John</first_name>
        <last_name>Doe</last_name>
      </observer>
      <location horizontal_datum="WGS84" vertical_datum="WGS84 ellipsoid">
        <latitude units="deg">35.85952</latitude>
        <longitude units="deg">-120.41479</longitude>
        <elevation units="m">120.0</elevation>
      </location>
      <obs_time>2004-09-30T20:52:00Z</obs_time>
      <photo_link>
        <src>http://www.usgs.gov</src>
      </photo_link>
      <fault>
        <name>San Andreas</name>
        <local_strike units="deg">150</local_strike>
        <local_dip units="deg">87</local_dip>
      </fault>
      <cracks_observed>true</cracks_observed>
      <slip_data>
        <slip_strike units="deg">095</slip_strike>
        <strike_angle_diff units="deg">55</strike_angle_diff>
        <slip_plunge units="deg">90</slip_plunge>
        <total_slip units="m">0.3</total_slip>
        <horiz_slip units="m">0.021</horiz_slip>
        <vert_slip units="m">0.001</vert_slip>
        <strike_slip units="m" uncertainty="0.001">0.012</strike_slip>
        <dip_slip units="m" uncertainty="0.001">0.012</dip_slip>
        <opening units="m" uncertainty="0.001">0.0</opening>
      </slip_data>
    </measurement>
  </measurements>
</surface_rupture>

```

Figure 5.2: Sample XML file for surface rupture models. Bold indicates element values, green, attribute names, and red, attribute values.

The scripts are available from the documentation section of the Reference Earthquakes Digital Library web site (<http://www.wr.usgs.gov>).

5.2.1.1 Converting to XML

We have provided an Excel template which can be used for entering data. From Excel, the information can be exported to a column formatted file with a choice of column delimiters. The script `surfrupture_xls2xml.py` will take this file and convert it to a surface rupture XML file. The user can specify the column delimiter (e.g., comma or tab) as well as the flag for missing information via command line arguments. For more detailed information about the command line arguments, run the script with the `-h` flag. Note that although the XML schema permits multiple observations at the same location, currently the script `surfrupture_xls2xml.py` treats each row of the input file as a new location with one observation.

5.2.1.2 Converting from XML

The script `surfrupture_xml2txt.py` will extract information from a surface rupture XML file and write it out in columns. This script is very similar to `seiscatalog_xml2txt.py`. The script uses command line arguments to specify the names of the input and output files (run the script with the `-h` flag for more information). The functions `writeHeader` and `writeEvent` define the output header information and the format of each event. More detailed information about setting up these functions can be found in the script.

6 Instrumentation

The digital library stores metadata associated with seismic and geodetic instrumentation in the immediate area of the reference earthquakes. The metadata includes information such as location, date of operation, and type of instrumentation. This information should enable researchers to find the data they need for their models. The instrumentation metadata are added by the digital library administrators. At some point in the future, it may become possible to harvest the instrumentation metadata from data centers in an automated fashion.

6.1 Seismic Instrumentation Metadata

The complete set of metadata for the seismic instrumentation includes the reference earthquake, canonical ADO, and geographic extent, as well as the seismic instrumentation-specific metadata components. The seismic instrumentation metadata components are shown in table 6.1.

Table 6.1: Seismic Instrumentation Metadata

Name of field	Brief Description
*Station ID	Station ID
*Station Name	Verbose station name
*Network	Network to which station belongs
*Channels	Channels in SEED data format
*Begin Date	Date when station started operating (YYYY-MM-DD)
*End Date	Date when station stopped operating (YYYY-MM-DD)
*Data Center Name	Name of data center
*Data Center URL	URL of data center
*Required	
#Optional	

6.2 Geodetic Instrumentation Metadata

The complete set of metadata for the geodetic instrumentation includes the reference earthquake, canonical ADO, and geographic extent, as well as the geodetic instrumentation-specific metadata components. The geodetic instrumentation metadata components are shown in table 6.2.

Table 6.2: Geodetic Instrumentation Metadata

Name of field	Brief Description
*Station ID	Station ID
*Station Name	Verbose station name
*Network	Network to which station belongs
*Components	Components observed (e.g., East, North, Up)
*Begin Date	Date when station started operating (YYYY-MM-DD)
*End Date	Date when station stopped operating (YYYY-MM-DD)
*Sensor Type	Continuous or campaign GPS, creepmeter, strainmeter, etc.
#Sensor Depth	Depth of sensor (positive down)
#Sampling Rate	Data sampling rate or approximate frequency if campaign
*Data format	Format of data (raw, RINEX, etc.)
#X Geocentric Location	Location in meters, specify reference frame
#Y Geocentric Location	Location in meters, specify reference frame
#Z Geocentric Location	Location in meters, specify reference frame
#Reference Frame	Reference frame of archived data if processed
*Data Center Name	Name of data center
*Data Center URL	URL of data center
*Required	
#Optional	