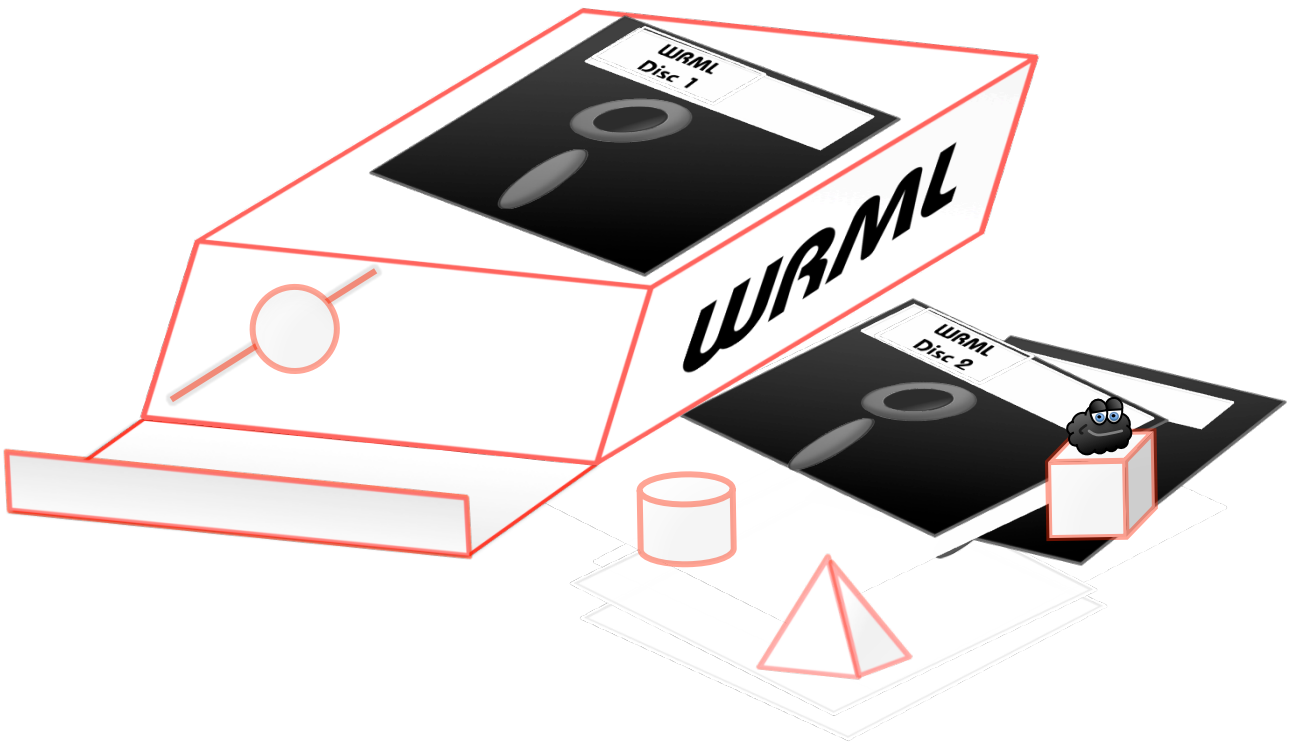
WERMINAL MASTER'S HANDBOOK

Version 1.0

by

Wormle, the Worm Wizard



The Map

Welcome to the **Werminal Master's Handbook**. This book helps guide weary travelers through Werminal, WRML's terminal-based data modeling tool.

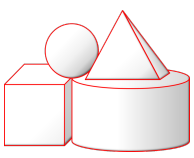
Below is the application's Map, with a box per screen and arrow to indicate available work flows.

You are here

Activating the Werminal Help will bring you to this documentation.

To learn more...

This document relates to WRML, the **Web Resource Modeling Language**. Check out WRML.org for more information.



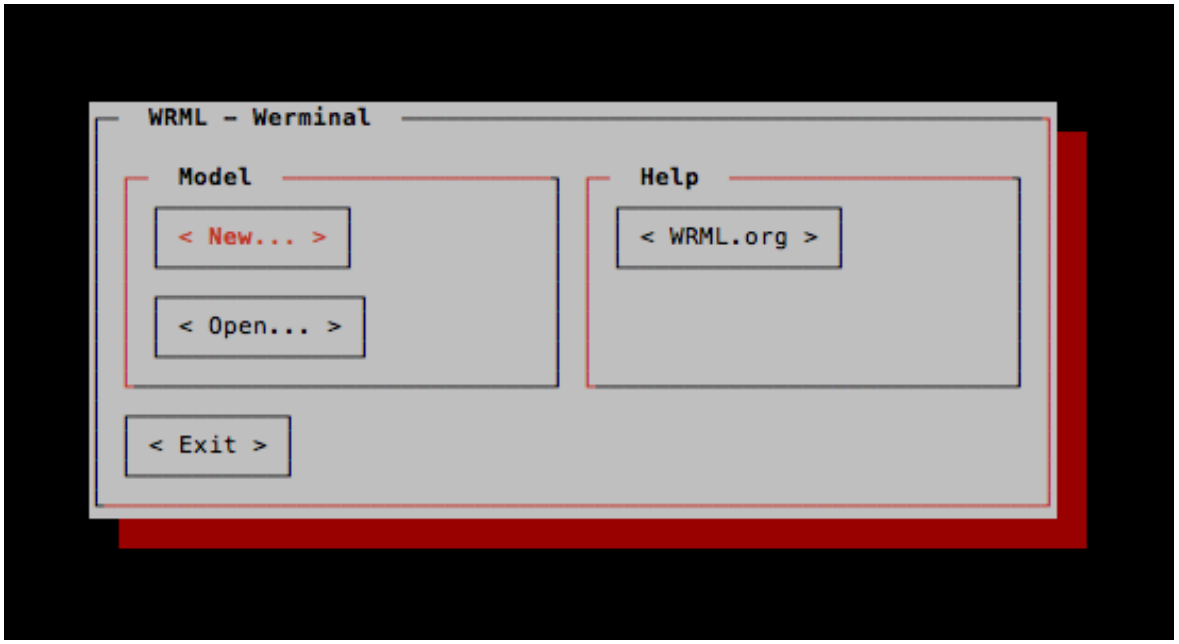
Splash Screen

Upon start-up, **Werminal** greets users with an introductory screen. The splash screen explains that Werminal is a **keyboard-driven**, terminal window based application.

To move things along, Werminal prompts users to “Press **any key** to start modeling”.

Main Menu

After the splash screen is dismissed (with any key), Werminal displays the Main Menu screen. This screen is home to the application's two primary features: **creating new models** and **opening existing models**.



< New... >

This action displays the New Dialog screen, which prompts the user to select a **Schema** for the new **Model**.

< Open... >

This action displays the Open Dialog screen, which prompts the user to select a **Schema** and enter **Keys** to identify the **Model** to be opened.

< WRML.org >

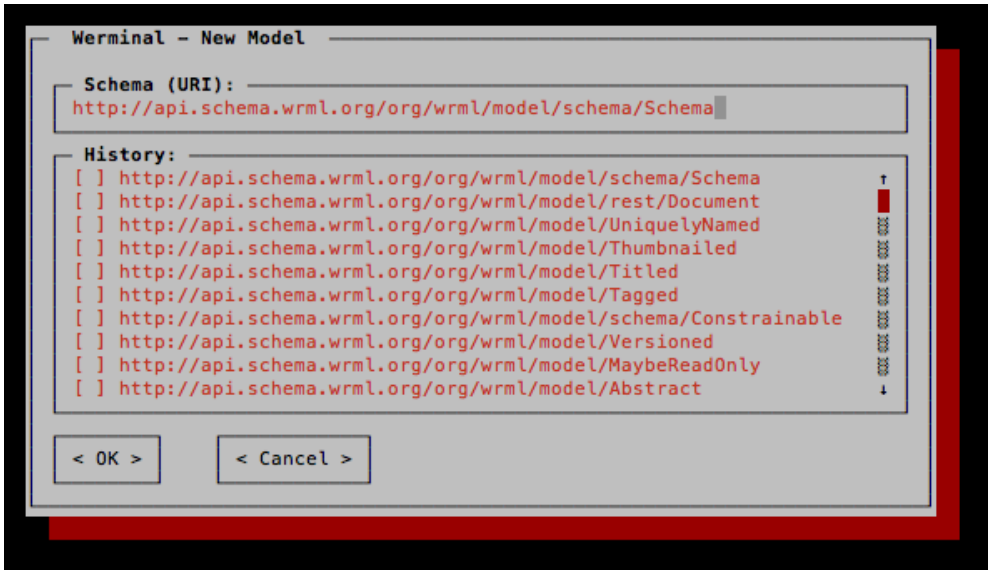
This action navigates a web browser app to this help documentation.

< Exit >

This action **quits** Werminal, returning control back to the parent console or terminal process.

New Dialog

The New Dialog screen asks users to decide what type of data model that they wish to create. This decision is made by typing, or selecting from the History, the URI of the Schema to be instantiated (with the option of saving/persisting the new model later).



Schema URI

The New Dialog's **Schema** text field allows users to enter a URI value to identify the type of the new **Model** to create. A Schema is metadata document that describes a set of named "slots" (name/value pairs) that may be present in any Model conforming to the Schema. A Model is an *instance* of a Schema. *NOTE:* As a REST-oriented system, WRML's Schemas are inherently multifaceted with representation as JSON, XML, Java, SQL, etc.

Schema URI History

Recalling and typing URIs is not very fun. The history list helps make the Werminal program more user friendly by asking humans to only type a given Schema URI once. Pressing **Space** or **Enter** checks the box in the selected row of the history list and sets the URI as the Schema text field's value. *NOTE:* Hitting the **Enter** key again on the selected and checked row will trigger the < OK > action.

< OK >

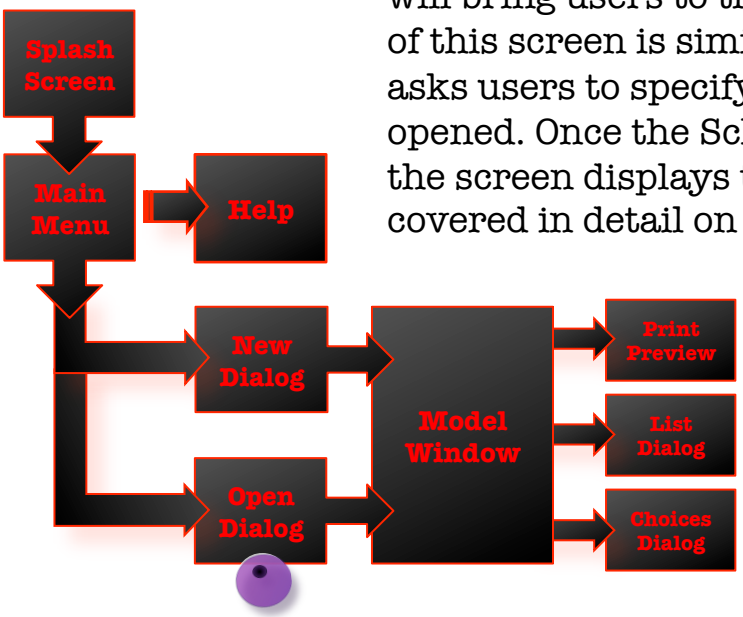
This action confirms the choice of Schema and displays a Model Window screen with the new (blank) model ready to be filled in and (optionally) saved.

< Cancel >

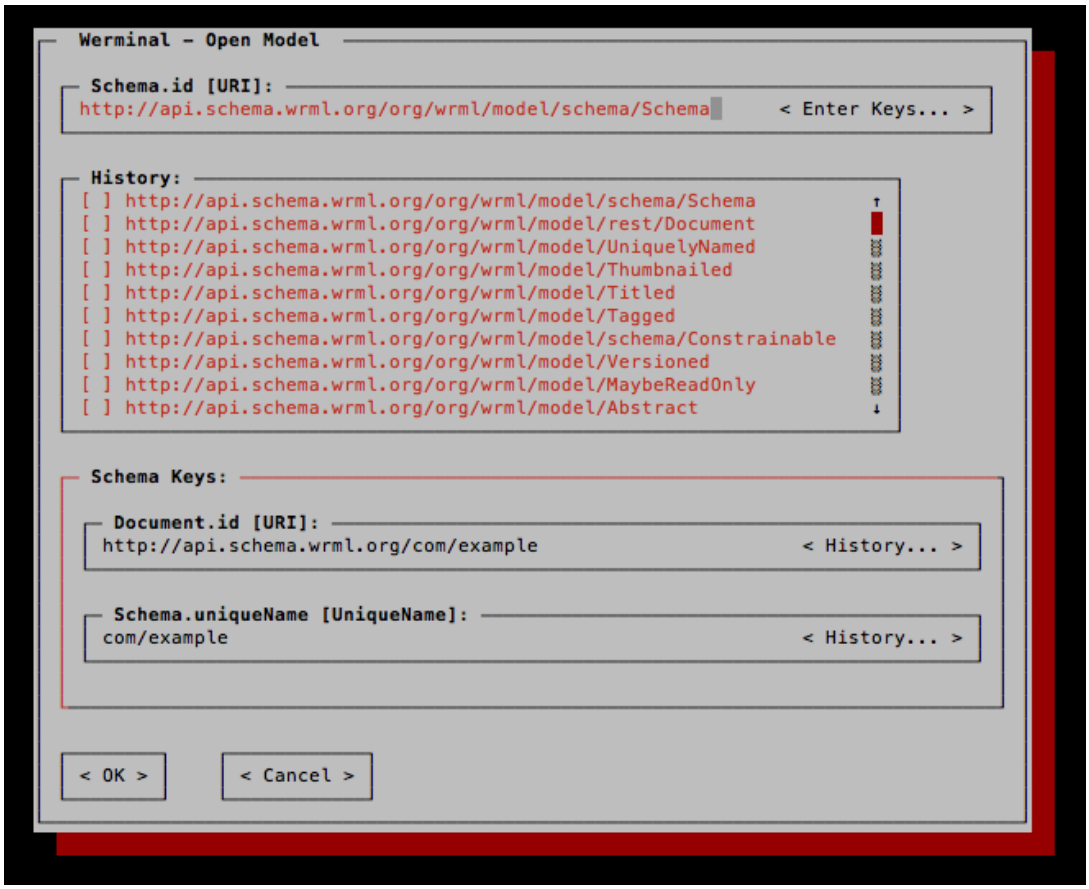
This action returns users to the Main Menu screen.

Open Dialog (part 1)

Triggering the < Open > action from the Main Menu will bring users to the Open Dialog screen. The bulk of this screen is similar to the New Dialog screen. It asks users to specify the type of model to be opened. Once the Schema URI has been entered, the screen displays the **Keys** entry form, which is covered in detail on the next few pages.



You are here



Open Dialog (part 2)

Opening Models with Keys

The lower portion of the Open Dialog features the key entry form area. This form is used to identify the model instance to be opened. The form dynamically arranges itself to provide the Schema-appropriate text fields needed to enter each possible key value.

Since **Keys** are an integral part of WRML, they warrant a proper introduction.

Modeling Uniqueness with WRML Keys

The WRML system uses Keys to represent the idea of IDs (identities) and to formalize the concept of identity *domains* or *spaces*. WRML allows clients to "open" (aka GET) model instances by one or more key values, which are opaquely collected together as Keys.

As a composition-oriented system WRML, aligns the notion of an identity space, such as the Web and its URIs, or the *Universe* and its UUIDs, with the concept of Schema *inheritance*. As a result, each WRML Schema may declare a key (thus declaring itself an identity space) by simply naming one (or more) of its own slots and/or inherited base Schema(s) slot(s).

Example Key Declaration

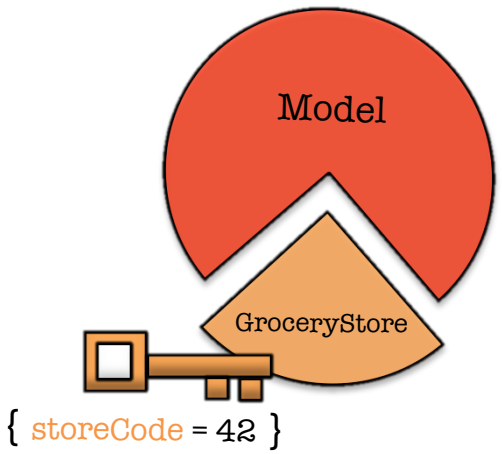
A **JSON**-formatted *GroceryStore* Schema declares its key slot(s) like this:

```
"keySlots" : [ "storeCode" ]
```

When represented as a **Java** interface, the same WRML Schema declares its key slot(s) using the Key annotation, as shown below:

```
@Key(slots = { "storeCode" })
public interface GroceryStore
```

Both express that *GroceryStore* model instances may be uniquely identified (and *indexed*) by their storeCode slot's value.



Open Dialog (part 3)

Key Inheritance

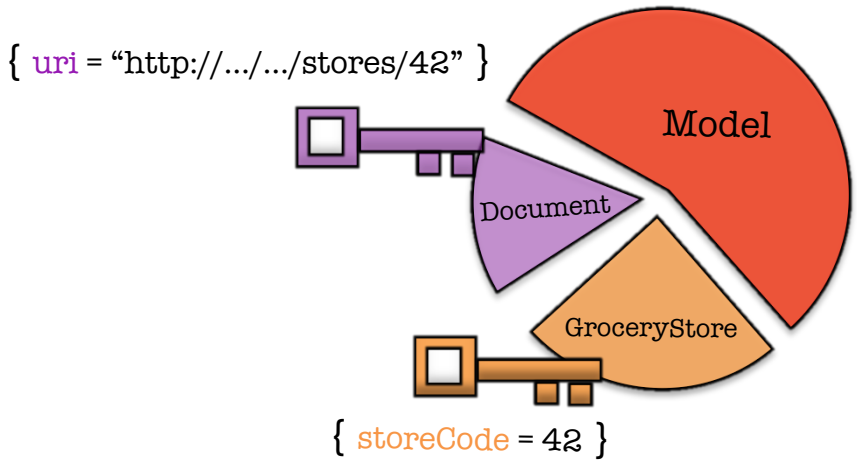
To participate in a given identity space, a Schema simply claims the appropriate Schema as a basis (a base Schema) for itself. For example, if models of a given Schema should have unique identities on the Web (e.g. as a REST API resource representations) then the Schema simply declares the **Document** Schema as a basis. As a result, its model instances will inherit the Document Schema's declared Key slot, *uri*.

Similarly, if a model instance should be considered universally unique, with a UUID-based identity, then it's Schema lists the **UniquelyIdentified** Schema as a basis and it gains the UUID-valued *uniquelyIdentifed* key slot.

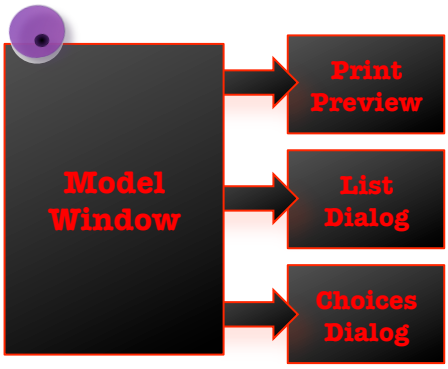
The multiple interface inheritance nature of WRML base Schemas means that a given model may have zero or more keys that function as separate, possibly related identifier values.

Links and URIs

For REST API-oriented (top-level) **Document** models, WRML's runtime automatically manages the translation of "internal" (surrogate) identity space identifiers (e.g. DB ids) to a REST API's identity space. In other words, WRML automates the **Link href** values by leveraging the combination of key slot values and REST API design metadata.



Model Window (part 1)



The Model Window is the Werminal application's primary screen.

By leveraging the WRML core engine and Schema metadata, this screen facilitates viewing, editing, and saving data models of any type.

The screen shot below demonstrates that Werminal can also be used to open *system models* such as the built-in Format REST API's model.

```
Werminal - Model - API - Format API
< Close >   < Print Preview >   < Save >

Schema:     http://api.schema.wrml.org/org/wrml/model/rest/Api
Heap ID:    e82a6031-3a3d-4adf-a3cb-d4d8a214476b

Slots
< Next >    1 of 2

id :
http://api.format.wrml.org

cacheTag :

delete :
href=""

description :
The WRML System Format REST API.

docroot :
8b45cde2-c846-449c-9df4-5772cfa77926 < Open... >

linkTemplates :
List<org.wrml.model.rest.LinkTemplate> [ 3 elements ] < Open... >

save :
href=""

secondsToLive :
0

self :
href=""

title :
Format API
```

Model Window (part 2)

Slot Pages

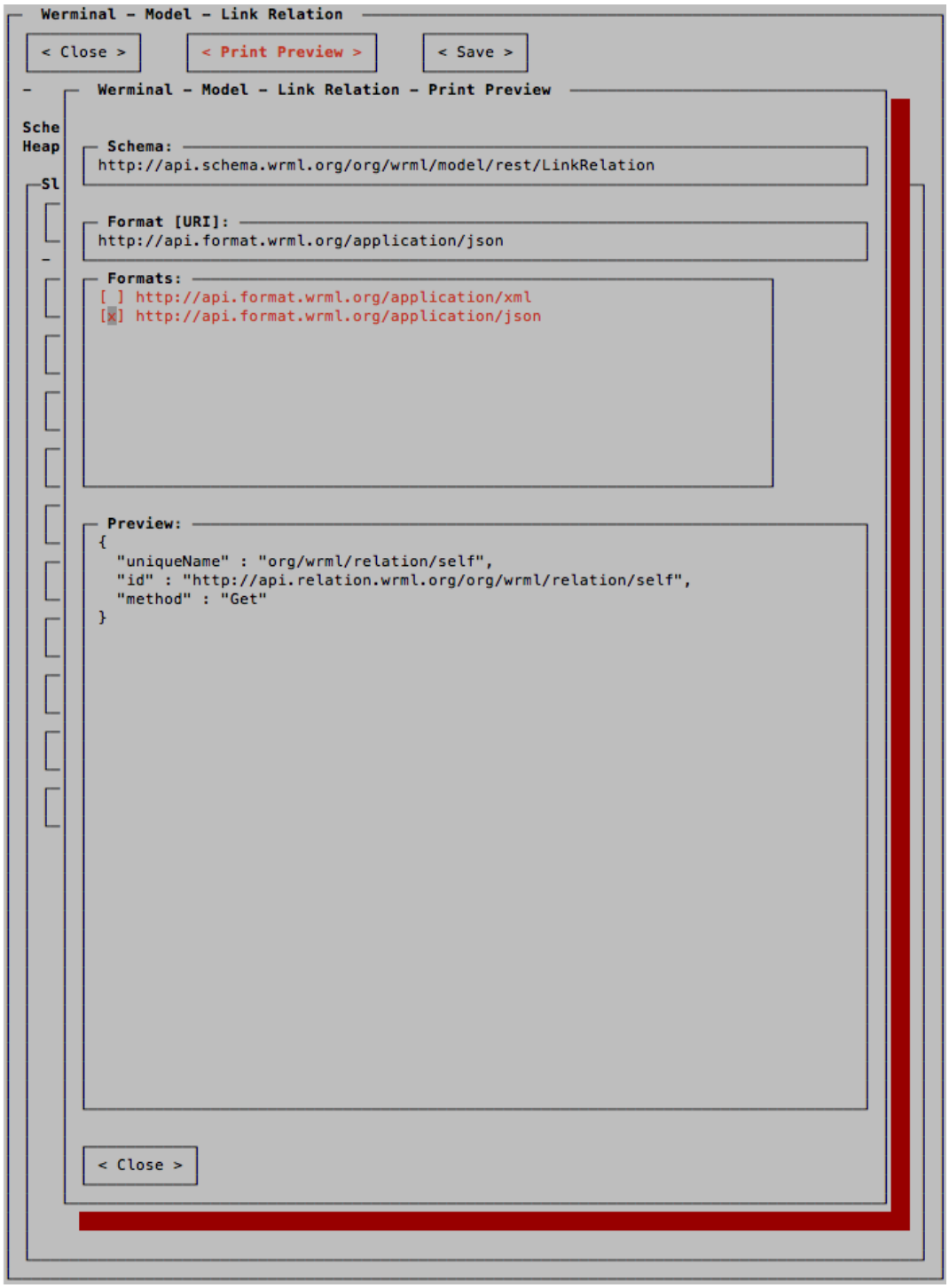
The Model Window's central feature are the slot pages, which list the model's available slots in alphabetical order (with key slots ordered first). The **Tab** key cycles the input focus forward through the slots. **Shift + Tab** cycles the focus in reverse order. The **Enter** key will open the slot value type-specific editor as outlined in the table below.

Value Type	Value Format and Editor Support
Boolean	**Enter** toggles the value between *true* and *false*.
Date	Text format as defined by RFC 2616 (section 3.3.1).
Double	For decimal numbers like 3.12.
Integer	For integer values (no decimal point).
Link	**Enter** clicks the Link, opening the result in a new Model Window.
List	A linear sequence of any other Value Type (except List). Enter key to open List Dialog.
Long	For compatibility with long (large integral) values.
Model	For embedded models (models within models). **Enter** on a blank opens the Model Value Initialization Dialog. Enter on a model value will open it in a new Model Window.
Single Select	A text value drawn from a menu of choices (like an *enum*). **Enter** key for Choices Dialog.
Text	A string of characters (with optional syntax constraint).

Model Window (part 3)

< Print Preview >

The



Model Window (part 4)

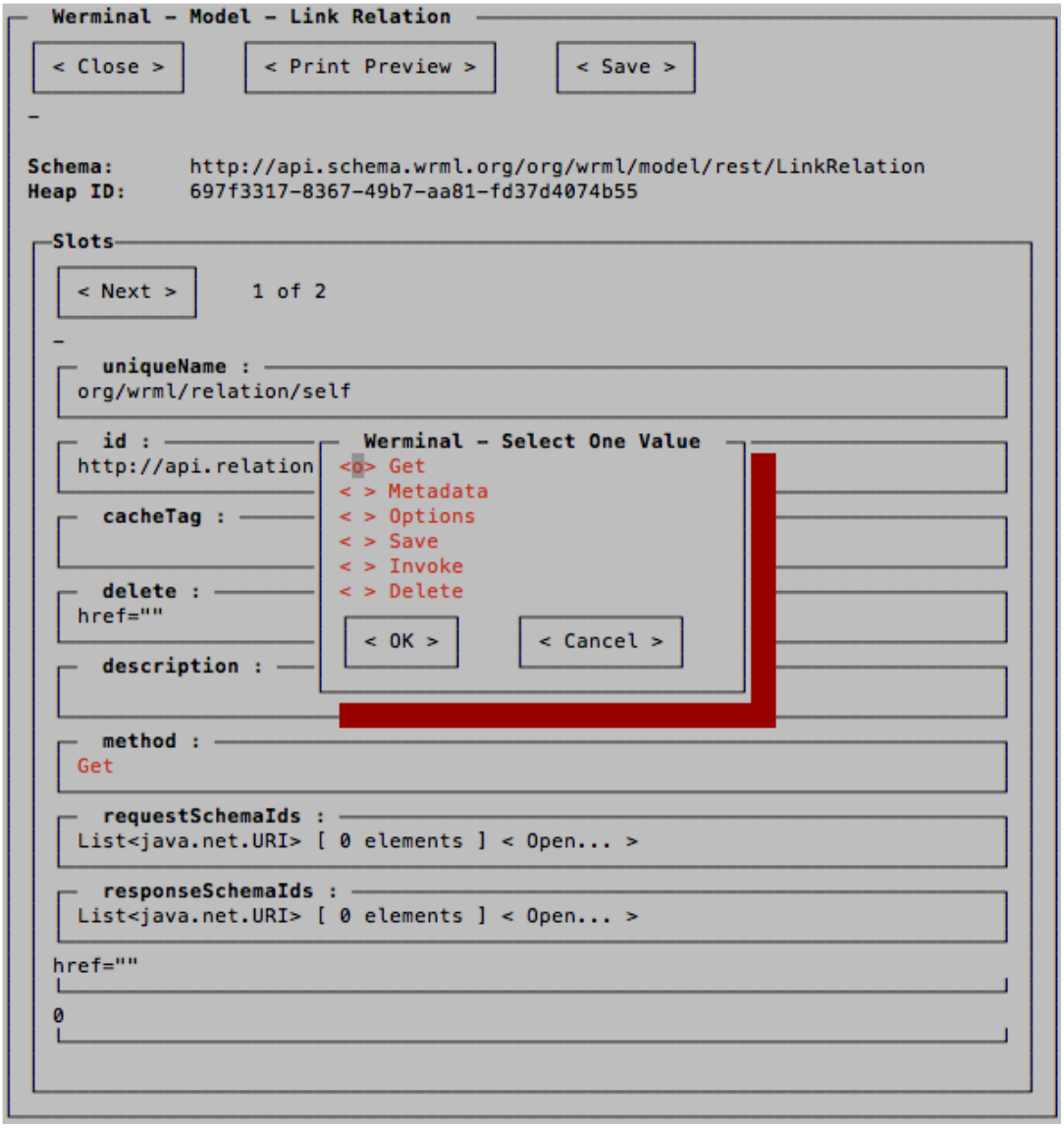
< Save >

The

Choices Dialog

Title

The Single Select Value...



List Dialog

Title

The

Model Value Initialization Dialog

Title

The