Preferences for Global Access

Institute for the Study of Knowledge Management in Education (ISKME)

Contract #ED-OSE-12-D-0013-0002

***Deliverable Task 2.2 Architecture Report***

***Cloud-Based Accessibility For Individuals with Disability***

Tools for Creating Specifications of User Needs and Preferences for Online Interactions in Several Different Aplication Settings

Prepared by:

ISKME

Ontario College of Art and Design University

IBM

WGBH

Raising the Floor International

Inclusive Technologies

The development of this document was led by PGA team member

Anastasia Cheetham

*Inquiries about this work can be addressed to lisa@iskme.org*

June, 2015

Table of Contents

[1. Introduction](#h.tgxwh8vee1z4)

[2. Preferences Framework](#h.damwx59t6wyy)

[Preferences](#h.llg0tvt02qmg)

[Declarative Configuration](#h.va9figyokvc6)

[Schemas](#h.d6uotksk42l9)

[Primary](#h.tbdfabwjt1uv)

[Auxiliary](#h.ofhp62rts5v7)

[Preferences Editor Loader definition](#h.ln1wrw4vfkbz)

[Localization](#h.cw7nf6pou4o8)

[3. First Discovery Tool Architecture](#h.i9wq1lt5rrwy)

[Structure of the First Discovery Tool](#h.8tcxbkalpnhh)

[Persistence](#h.j5my3us360xu)

[Template Management](#h.ywr9b360dju2)

[Message Loading](#h.q92wv8dpv6c4)

[Navigation](#h.vj0mx8xnfrmm)

[Self Voicing](#h.9a37r6e9sf1)

[UI Enhancement](#h.ym86m3v5ptvg)

[4. Contextualization](#h.gqjd9sjiszc7)

[Re-skinning](#h.9jqc3dzab635)

[Re-wording](#h.6p3r4a5bykxh)

[Reordering Preferences](#h.dpizrqhwn68)

[Removing Preferences](#h.6gb2tniedbck)

[Adding Existing Preferences](#h.nvs7jlgee5bz)

[Adding New Preferences](#h.2hvcyb3zx9sb)

[Changing Preference Ranges and Defaults](#h.jx1buo2etd9s)

[Changing the View of a Preference](#h.dqki3itp5pin)

[5. Next Steps](#h.wr57yt1hfhfg)

[Confirmation](#h.mjswvwwd5a4g)

[Preview](#h.tb3blt89kv0o)

[Integration with GPII Preferences Server and Flow Manager](#h.q4t7w894ki3t)

[Administrative Interface](#h.dc2bhe5woict)

[Contextual Help](#h.jo9dsfwomox6)

[Tool State Persistence](#h.f0xxwlisgexe)

[Appendix: Technical Information](#h.9eaq574auugz)

[Component Tree](#h.ym4y9uwlcufk)

[References](#h.uu9byqwmgycm)

#

# 1. Introduction

This document describes the architecture of the Preferences for Global Access (PGA) First Discovery Tool. The PGA project was funded by the National Institute on Disability and Rehabilitation Research (NIDRR), a component of the U.S. Department of Education's Office of Special Education and Rehabilitative Service, under Task Order 0002 of ED-OSE-12-D-0013 - *Tools for Creating Specifications of User Needs and Preferences for Online Interactions in Several Different Application Settings*. The purpose of Task Order 0002 was to develop and evaluate example First Discovery Tools that address the creation of user needs and preferences specifications for users in the following four application settings: accessible voting, online educational assessment, community-based technology support for older citizens, and open educational resources. The purpose of examining the four application settings was both to inform the design of the prototype tool, and to identify how the tool would be modified and integrated into different settings – including who would be doing the modifying.

The First Discovery Tool is an open-source, freely available JavaScript application. A [live prototype of the tool](http://build.fluidproject.org/first-discovery/demos/) is publicly available, and [the source code can be found in the project’s Github repository](https://github.com/fluid-project/first-discovery/).

[Section 2](#h.damwx59t6wyy) of this document describes the *GPII Preferences Framework*, upon which the First Discovery Tool is built. [Section 3](#h.i9wq1lt5rrwy) explains how the First Discovery Tool is constructed using the Preferences Framework. [Section 4](#h.gqjd9sjiszc7) describes how the Tool can be integrated into different contexts, and [Section 5](#h.mvrgpjzgempf) provides an overview of our plans for future development of the First Discovery Tool.

# 2. Preferences Framework

The First Discovery Tool uses the *GPII Preferences Framework*, a reusable set of schemas, APIs, and user interface building blocks that is employed in the development of each of the preference tools in the [Global Public Inclusive Infrastructure (GPII)](http://gpii.net/)[[1]](#footnote-1). The Preferences Framework was a key development of Task Order 0001 of the Preferences for Global Access project (Petrides et al., 2013b).

## Preferences

The Preferences Framework models a ‘preference’ as three discrete objects, illustrated in Figure 1 and described below.



Figure 1: Anatomy of a Preference

* **View**
The view component is responsible for specifying the markup, styling, and template required to render a *panel* – the part of the interface containing controls that allow the user to modify a preference value. A control is referred to as an *adjuster*.
* **Enactor**
An enactor does the actual work of accommodating the user's preference. The complexity of an enactor will depend on what it is doing: some enactors may be relatively simple (for example, changing the text size on a web page); some enactors may be quite complex, even delegating the actual work to another module (for example, implementing text-to-speech on a page).
* **Persistence** information
Persistence information includes schematic information about the preference – its value types, ranges and default values – as well as a *data store* component which connects the preference editor to a configurable form of storage for recording preferences.

The view, enactor and persistence information for a preferences editor is provided in configuration files described in the next section.

## Declarative Configuration

The Preferences Framework and the First Discovery Tool architecture follow [Fluid Infusion](http://fluidproject.org/infusion.html)‘s declarative [Inversion of Control (IoC)](http://www.martinfowler.com/articles/injection.html) philosophy: With Infusion IoC, instead of writing code with tightly-coupled dependencies, developers express their application structure as a declarative “component tree” that is instantiated and managed by a context-aware framework (Basman et al., 2011). This makes it easier for third-party developers and designers to refine and extend a preference editor over time, and to contextualize and integrate it into any environment (see the [Integration and Contextualisation](#h.wivqwym6oa9m) section below for information about how the First Discovery Tool can be customized).

The Preferences Framework’s *Builder* component constructs a preferences editor based on declarative configuration information provided in

* a Primary Schema,
* an Auxiliary schema, and
* a Preferences Editor Loader definition

as illustrated in Figure 2. Each of these is described below.



Figure 2: Preferences Framework Builder workflow

### Schemas

#### Primary

The Primary Schema is a JSON document that contains the information necessary to define a preferences editor’s preferences using the format specified by the [JSON Schema specification](http://json-schema.org/documentation.html). The schema defines such things as the type of the preference, its default value, the limits of its range (if appropriate), an enumeration of possible values (if appropriate), etc.

The following example from the First Discovery Tool illustrates the structure of the schema (the full Primary Schema for the First Discovery Tool can be found in [the source code for the project](https://github.com/fluid-project/first-discovery/blob/master/src/schemas/schemas.js). ):

|  |
| --- |
| fluid.defaults("gpii.firstDiscovery.schemas.language", { gradeNames: ["autoInit", "fluid.prefs.schemas"], schema: { "gpii.firstDiscovery.language": { "type": "string", "default": "en-US", "enum": ["en-US", "fr-FR", "es-MX", "de-DE", "nl-NL", "sv-SE"] } }});fluid.defaults("gpii.firstDiscovery.schemas.speak", { gradeNames: ["autoInit", "fluid.prefs.schemas"], schema: { "gpii.firstDiscovery.speak": { "type": "boolean", "default": true } }});fluid.defaults("gpii.firstDiscovery.schemas.textSize", { gradeNames: ["autoInit", "fluid.prefs.schemas"], schema: { "fluid.prefs.textSize": { "type": "number", "default": 1, "minimum": 0.2, "maximum": 1.2, "divisibleBy": 0.1 } }}); |

#### Auxiliary

The Auxiliary Schema is a JSON document that defines the information needed to build a preference editor’s interface, including

* the preferences editor loader component to manage the entire user interface,
* component(s) needed to render the preference panels,
* locations of HTML templates and string bundles,
* component(s) needed to act on preference settings.

The following example (taken from the First Discovery Tool) illustrates the structure of the schema (the full Auxiliary Schema for the First Discovery Tool can be found in [the source code for the project](https://github.com/fluid-project/first-discovery/blob/master/src/schemas/schemas.js)):

|  |
| --- |
| { "loaderGrades": ["gpii.firstDiscovery.firstDiscoveryEditor"], "namespace": "gpii.firstDiscovery", "templatePrefix": "../src/html/", "template": "../src/html/firstDiscovery.html", "messagePrefix": "../src/messages/", "message": "%prefix/firstDiscovery.json", "textSize": { "type": "fluid.prefs.textSize", "enactor": { "type": "fluid.prefs.enactor.textSize" }, "panel": { "type": "gpii.firstDiscovery.panel.textSize", "container": ".gpiic-fd-prefsEditor-panel-size", "template": "%prefix/rangeTemplate.html", "message": "%prefix/textSize.json" } }} |

### Preferences Editor Loader definition

The Preferences Framework’s *Preferences Editor Loader* component coordinates the collaboration between the preferences editor, the template loading functionality and the message loading functionality (described below). The Loader also configures other functionality of the preferences editor, such as navigation and self-voicing in the case of the First Discovery Tool. This functionality is described in [Section 3](#h.i9wq1lt5rrwy).

## Localization

The Preference Framework supports localization through the use of message bundles: JSON files containing the strings to be used in the interface. Message bundles are JSON files containing key/value pairs representing the message key and the localized text associated with it. Each set of localized text is contained in its own message bundle. The Preference Editor loads the appropriate message bundles based on the requested language and makes the bundles available to the panels for rendering.

# 3. First Discovery Tool Architecture

## Structure of the First Discovery Tool

A conceptual representation of the structure of the First Discovery Tool is presented in Figure 3. Each of the areas of functionality is described below.



Figure 3: Conceptual structure of the First Discovery Tool

and how the declarative information shapes it

*In this diagram, the dashed lines indicate that the source provides*

*information that configures the destination*

### Persistence

The First Discovery Tool uses a component called a *Settings Store* (offered by the Infusion Preferences Framework) to manage the persistence of the user’s preferences. Currently, the First Discovery Tool is configured to use the Cookie Settings Store, which saves the preference values in a browser cookie. This is a temporary measure. In the future, the GPII toolset will provide a Settings Store that will communicate directly with the GPII preferences server to record the user’s preferences, ensuring privacy and security.

### Template Management

Individual portions of the First Discovery Tool interface have their own HTML templates. The Preferences Framework coordinates the loading of the template files and their use by the relevant Infusion component.

### Message Loading

All of the text in the First Discovery Tool interface is stored in JSON files called “message bundles.” A sample message bundle is shown below:

|  |
| --- |
| { "instructions": "Do you want to see text for speech (captions) when playing videos?", "no": "No", "yes": "Yes", "yes-tooltip": "Select to turn video captions on", "no-tooltip": "Select to turn video captions off", "yes-tooltipAtSelect": "Video captions are on", "no-tooltipAtSelect": "Video captions are off"} |

Message loading functionality in the Infusion Framework is used to load the relevant strings into the interface. Internationalization is accomplished by providing alternate JSON files containing translated strings and directing the message loader to load the correct files.

### Navigation

The First Discovery Tool presents preferences to the user one at a time. The user navigates from one preference panel to the next linearly, using ‘next’ and ‘back’ buttons. The component that manages the navigational buttons responds to activation of the buttons by updating the tool’s internal model of the “current” panel. The preference editor component responds to the change in model by presenting the appropriate panel.

### Self Voicing

The First Discovery Tool uses the browser-based [Web Speech API](https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html)[[2]](#footnote-2) to speak the interface aloud. While the implementation is not as thorough as that offered by a screen reader, it avoids any dependence on third-party applications, and does not require any understanding of how to interact with screen readers, which often require particular key sequences and other interactions that take time to learn. Providing text-to-speech as part of the First Discovery Tool ensures that users who would benefit from the functionality will be able to participate in the First Discovery process.

### UI Enhancement

Several of the First Discovery Tool preferences have an immediate effect on the interface of the tool itself. A UI Enhancer component adds enactors to the tool to carry out these changes. This implements users’ preferences immediately, supporting the user in aligning the First Discovery Tool with their preferences as they use the tool. The enactors that will respond to the preferences outside of the First Discovery Tool context may be slightly different, but the First Discovery Tool provides the user with an idea of how the preference will affect their experience with other programs and devices.

# 4. Contextualization

This section describes the technical requirements for modifying the First Discovery Tool, a task that might be performed by a staffer of an organization that has adopted GPII, such as a school district technology coordinator or an election office system administrator. There is currently no user interface (see [Administrative Interface](#h.dc2bhe5woict) in the following section) for this process, but these customizations typically require only basic web skill such as HTML, CSS, and JSON. This section describes the underlying modifications required for these contextualizations. This section focuses on contextualizations that are currently possible with the code.

A contextualization of the First Discovery Tool will likely fall into one or more of the following categories:

## Re-skinning

A basic form of contextualization of the First Discovery Tool involves changing the visual appearance of the tool, or “skinning” it. For example, a community centre may wish to provide their own colour scheme, logo and introductory page.

Actions required:

* modify/override CSS styles
* *(optional)* modify HTML templates

## Re-wording

A simple form of contextualizing the First Discovery Tool involves changing the text presented in the interface. For example, an primary school might wish to simplify the language of the tool when it will be used by young children.

Actions required:

* edit JSON message bundles

***Note****: the message bundles for each of the corresponding languages should also be updated as needed.*

## Reordering Preferences

The basic First Discovery Tool presents preferences in an order that is felt to be appropriate for most cases, but it may happen that for a particular integration, a different order of presentation may be more appropriate. In this case, contextualization would involve reordering the preferences.

Actions required:

* edit the firstDiscovery.html template to reorder placeholders
* edit index.html to reorder icons

## Removing Preferences

In some cases, such as a voting kiosk, it will make sense to remove many of the panels if they are irrelevant to the context. In a voting kiosk context, preferences would not be saved for later use, but would only apply for the current voting session, and so should be limited to preferences relevant for the capabilities of the given kiosk. For example, if the interface for the kiosk is in black and white and can’t be altered, then it would be reasonable to remove the colour contrast preference.

Actions required:

* remove the panels’ configuration from the auxiliary schema
* edit the firstDiscovery.html template to remove placeholders
* edit index.html to remove icons

## Adding Existing Preferences

In the future, the First Discovery Tool may include preference panels that are not, by default, included in the interface. Contextualization in this case might include adding some of these existing preference panels to the interface.

##

Actions required:

* add the panels’ configuration to the auxiliary schema
* edit the firstDiscovery.html template to add placeholders
* edit index.html to add icons

## Adding New Preferences

Some contexts may have features or technologies that could be adapted to an individual’s preferences in ways not presented by the First Discovery Tool. For example, an educational institution might use a particular notification system, and may wish to offer students preferences specific to how notifications are delivered. In these cases, contextualization could include adding new panels with relevant preferences.

Actions required:

* add primary schemas for the new preferences
* add new panel components *(may be able to extend an existing one)*
* add new HTML templates for the new panels *(may be able to reuse an existing template)*
* add JSON message bundles for the new panels *(in each language)*
* update the auxiliary schema with configuration for the new panels
* edit the firstDiscovery.html template to add new placeholders for the new panels
* edit index.html to add icons for the new panels
* update CSS to add the icon classes *(the First discovery tool uses icon fonts which are inserted via CSS)*

## Changing Preference Ranges and Defaults

One form of contextualisation could involve changing the possible values for a preference. For example, the range for a preference might have particular limits in a given integration, or a different default value might be desired. If the preference is a range or list of choices, and the intention is to maintain the same presentation form (e.g. keeping a slider in a panel that already includes a slider), then modifying the tool requires only that the defaults be changed.

Actions required:

* edit the preference definition in the primary schema

## Changing the View of a Preference

In some cases, changes to the type of a preference may lead an integrator to choose a different interaction for modifying the preference. For example, if a voting kiosk reduces the range of text sizes to only three (small, medium and large) because of the limits of the technology, the integrator may want to replace a slider with a set of radio buttons to simplify the user experience and to reduce motor control dependencies.

Actions required:

* edit the preference definition in the primary schema
* modify HTML templates for the panels to reflect new adjuster
* modify JSON message bundles to reflect new adjuster labels
* edit the JavaScript for the panel to reflect the new rendering
* update the auxiliary schema with configuration for the preference type

# 5. Next Steps

## Confirmation

One key requirement identified early on is the need for the user to be able to confirm all of their preference choices before completing the First Discovery process. This step is not yet implemented in the current prototype.

## Preview

Currently, some of the preferences (such as text size and colours) directly alter the appearance of the First Discovery Tool itself. This is intended to help the user understand how the preference will affect what they see, but it may not adequately prepare the user for how their preference choices will affect other content. A Preview would support users in understanding how their preference choices might affect content outside of the First Discovery Tool. The Preview would present a sample of content typical in the application area, and preference choices would be applied to the preview to help the user understand how their choices would affect what they will see after the First Discovery process. The design team has started working on ideas for how a Preview could be incorporated into the interface.

## Integration with GPII Preferences Server and Flow Manager

The Preferences for Global Access architecture is, as described above, closely aligned with the overall Global Public Inclusive Infrastructure. As part of this alignment, a robust and comprehensive architecture has been designed to support the personalization of desktop, mobile, and web applications (Clark et al., 2104). In future versions of the First Discovery Tool, users will be able to explore and discover preferences and adaptations not only for web-based content but also native platform tools such as assistive technologies like screen readers, screen magnifiers, and operating-system-level keyboard accessibility settings[[3]](#footnote-3). Further information about the GPII's architecture for web, desktop, and mobile personalization, including the configuration of assistive technologies, is available in the [GPII wiki](https://wiki.gpii.net/w/Architecture_Overview).

## Administrative Interface

An authoring interface will be developed to allow someone to customize and configure the tool without having to edit configuration files or source code directly.

## Contextual Help

A help system will be developed to provide context-dependent help, including guidance on how to use the tool as well as some basic digital literacy training.

## Tool State Persistence

Currently, the state of the tool does not persist, so if a user’s interaction with the tool is interrupted, they cannot yet “pick up where they left off.” We will be expanding the information that is recorded concerning the state of the tool and implementing persistence of this information. When the First Discovery Tool is integrated with the GPII preferences server, this may involve saving the tool state along with whatever preferences have been set so far.

# Appendix: Technical Information

## Component Tree

Figure A.1 provides a graphical representation of the internal component hierarchy of the First Discovery Tool. The table below describes the responsibilities of each of the components.



Figure A.1 The First Discovery Tool component hierarchy

*In this diagram, each box represents an* [*Infusion*](http://fluidproject.org/infusion.html) *Framework component*

|  |  |
| --- | --- |
| **Component** | **Responsibility** |
| firstDiscovery  | the Builder-assembled component for the entire tool  |
| store | persistence of preference values |
| settingsStore | specific interface for persisting preferences in a browser cookie |
| enhancer | page-level component scopes adjustments to the entire page |
| uiEnhancer | coordinating enactors |
| [enactors] | applying a preference to the current page |
| prefsEditorLoader | coordinates all the parts of the tool interface |
| templateLoader | loads HTML templates |
| messageLoader | loads string bundles |
| selfVoicing | text-to-speech; respond to mute button activation |
| helpButton | a placeholder component to render the text, and will eventually launch the tools help system |
| prefsEditor | coordinate panels |
| [panels] | render adjusters for preference values and respond to input by updating the model value for that preference |
| nav | container for navigational components |
| navButtons | render the next/back buttons and respond to activation by updating the model’s currentPanel value |
| navIcons | indicating status: current page and completed pages |
| stepCount | render the message “step X of Y” at the bottom of the tool |

# References

Basman, A., Lewis, C., Clark, C. “To inclusive design through contextually extended IOC”. In C. Videira Lopes and K. Fisher (eds): *Companion to the 26th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA 2011. pp. 237-256.

Clark, C., Basman, A., Bates, S., Markus, K. G. “Enabling architecture: How the GPII supports inclusive software development”. *Universal Access in Human-Computer Interaction. Design for All and Accessibility Practice*, Springer International Publishing, 2014. 368-377.

Petrides, L., McLaughlin, L., Jimes, C., Brennan, M., Treviranus, J., Schwerdtfeger, R., Rothberg, M., Vanderheiden, G., Tobias, J., Trewin, S., Clark, C., Mitchell, J., (2013a). *Preferences for Global Access Design Report: Profile Creation Support for Cloud-based Accessibility*. Unpublished manuscript, submitted to the U.S. Department of Education's Office of Special Education and Rehabilitative Service.

Petrides, L., Jimes, C., McLaughlin, L., Treviranus, J., Schwerdtfeger, R., Rothberg, M., Vanderheiden, G., Tobias, J., Trewin, S., Clark, C., Mitchell, J., and Roberts, V. (2013b). *Preferences for Global Access: Profile creation support for cloud-based accessibility*. Unpublished manuscript, submitted to the U.S. Department of Education’s Office of Special Education and Rehabilitative Services.

1. The First Discovery Tool is one of several preference editing tools developed as part of the larger GPII effort. [↑](#footnote-ref-1)
2. The Web Speech API is not yet supported in all browsers. See <http://caniuse.com/#feat=web-speech> for more information. [↑](#footnote-ref-2)
3. This is dependent on the completion of APIs that are currently being developed by the [Cloud4All](http://www.cloud4all.info/development/) project. [↑](#footnote-ref-3)