**United States Department of Commerce**

**National Telecommunications & Information Administration**



**System Design Document for BroadbandUSA Connectivity Initiative**

**(BCI)**

March, 2017

NTIA/OPCM/ITD

1401 Constitution Avenue, NW

Washington, DC 20230

**Table of Contents**

[1. Introduction 3](#_Toc478538414)

[2. Design Assumptions 3](#_Toc478538415)

[2.1 Assumptions about WordPress Content 3](#_Toc478538416)

[2.2 Assumptions about Teams 4](#_Toc478538417)

[3. System Architecture 4](#_Toc478538418)

[3.1 Hardware Environment 4](#_Toc478538419)

[3.2 Software Environment 5](#_Toc478538420)

[3.3 System Architecture 5](#_Toc478538421)

[3.4 Website Navigation 9](#_Toc478538422)

[Modules Defined in Version 1 10](#_Toc478538423)

[3.5 Network Considerations 11](#_Toc478538424)

[3.6 Deployment Considerations 12](#_Toc478538425)

[4. Database Design 13](#_Toc478538426)

[5. System Security 17](#_Toc478538427)

[5.1 Role-Based Access Control 17](#_Toc478538428)

[5.2 Component Access 18](#_Toc478538429)

[5.3 Auditing 18](#_Toc478538430)

[Appendix I. Terminology 19](#_Toc478538431)

[Appendix II: Custom Taxonomies 21](#_Toc478538433)

[Appendix III: Custom Post Types and Custom Post Metadata 23](#_Toc478538434)

[The Custom Post Types 23](#_Toc478538435)

[The Prompt CPT 23](#_Toc478538436)

[Prompt Content 24](#_Toc478538437)

[Shortcodes for Data Items 24](#_Toc478538438)

[Prompt Navigation 26](#_Toc478538439)

[Post Data and Metadata used for Prompts 26](#_Toc478538440)

[Representing Prompt Answers in WordPress Post Metadata 26](#_Toc478538441)

[The Response Custom Post Type 27](#_Toc478538442)

[Wordpress Data attributes for a Response. 27](#_Toc478538443)

[Response Post Metadata Usage 27](#_Toc478538444)

[The Resource (Recommendation) Custom Post Type 28](#_Toc478538445)

[Resource Post Metadata Usage 28](#_Toc478538446)

[The Report Element CPT 28](#_Toc478538447)

[Appendix IV: Configuration Options Defined in Version 1 30](#_Toc478538448)

[Appendix V: BCI User Roles and Privileges 30](#_Toc478538449)

# Introduction

The BroadbandUSA Connectivity Initiative (BCI) system aims at the users who wish to accelerate their broadband connectivity planning for their geographical region. An assessment consists of a collection of responses to a series of questions on Broadband connectivity plan for a given location. The system supports assessment for multiple locations. It also incorporates multiple data set and resources such as Census shapefile data, FCC tabular data and NTIA CPS data to provide recommendations for improvement of the assessment. The result of an assessment is provided in a format of reports.

This document describes the architecture for the BCI system. Appendix I. Terminology defines the terms used throughout the document.

# Design Assumptions

### **2.1 Assumptions about WordPress Content**

1. A Module consists of an ordered series of Prompts.
2. An actionable Prompt can have a zero or more can have zero or more Responses. Actually, a Prompt is actionable if a Response other than ‘OK’ is allowed. Prompts that simply present information *are not* considered actionable.
3. A Response represents one user’s reply to an actionable Prompt.
4. A user can provide only one single Response to a Prompt (one user, one Response).
5. Users may edit their Response at any time.
6. Each actionable Prompt can have only one single Team Response. Team Responses form the basis for resource recommendations. Team Responses can be edited by any user with ‘Team Editor’ privileges. In our user-model, this includes Team Editors and Team Leads. This implementation tempers the one-user, one-Response rule since each Team Editor can provide their own personal Response, and help to edit the Team’s final Response.
7. Each Prompt has a well-known result type (e.g. yes/no or check-all-that-apply). The response types are defined by the approved content, and are implemented by the Plugin. This means that adding a new response type means a minor code edit in the current code base.
8. Prompts can contain arbitrary (formatted) text, and can pull in data tables and other visualizations that illustrate the local broadband situation.
9. At any stage of completion, users can view the Responses for a Module.
10. Resources describe possible additional content that a team might wish to consider or implement, based on their Team’s Responses to a Module. Internally, resources are linked to the possible Responses of a given Prompt, so they can be very specifically targeted.
11. Since the content has had to pass multiple rounds of reviews, the Modules, Prompts, and Response-Types are well-known, and can be prepopulated using bulk loading. More importantly, interactively adding new alternatives to multiple-choice Responses is not supported in the current version. This was an implementation choice.
12. The Version 1 prototype supports ‘follow-on’ Prompts that are only displayed base. This feature is not used in the Version 1 approved set of Prompts.
13. In order to provide “national data localized” to team members, it should be easy for non-technical editors to include data elements.

### **2.2 Assumptions about Teams**

1. A team consists of one or more individuals who are motivated to use the BCI tool to accelerate their broadband planning.
2. The members of a team will cooperate to fill out their responses in single collection of Modules and Prompts, called a “Workbook”
3. Team members consist of individuals from outside NTIA. Presumably, they will be state- or local-level contributors.
4. Each Team has one or more designated leaders who can be made responsible for the composition of the team. Specifically, team leads could invite other persons to join the team, and can assign roles to each person on the team. This feature is currently disabled in Version 1.
5. Team Leads may or may not wish to play the role of Team Editors; that is, the Team Leads may not be entirely responsible for crafting their team’s Responses to Prompts. For that reason, the role of “Team Editor” is also implemented.
6. Teams may include persons who are asked to contribute individual Responses.
7. Teams may include persons who are able to read Responses (and comment on them), without being able to construct Responses. In the current prototype, commenting has not yet been deployed for testing.
8. In Version 2 of the tool, local- and state- Broadband Planning partners may want to participate in multiple Teams, and be able to see status information across all of their teams. This role “Community Partner” is limited in Version 1 to participating like a “Community Participant”.
9. NTIA users need access to edit content (e.g. Prompts) to support the Team leads (e.g., update the Plugin or reset passwords), or just to keep a weather eye on the entire process. NTIA-specific roles include “Assessment Editor”, the normal WordPress Administrator and Network Administrator roles, and an “Assessment Manager” role. The “Assessment Manager” role has not been implemented in Version 1.

# System Architecture

### **3.1 Hardware Environment**

The system prototype is currently installed on the cloud service cloud.gov, which consists of four servers:

1. Web server. It contains httpd server, WordPress, WordPress CLI, WordPress CLI scripts, BCI Tool as custom code (Theme & plugin).
2. MySQL server for WordPress
3. PostGIS database server
4. Data server: which server does this Perl-based code reside

The current NTIA testing environment uses one Linux server (Red Hat 6) for all BCI components.

### **3.2 Software Environment**

1. Open source software

The current BCI system is built on the following open source packages:

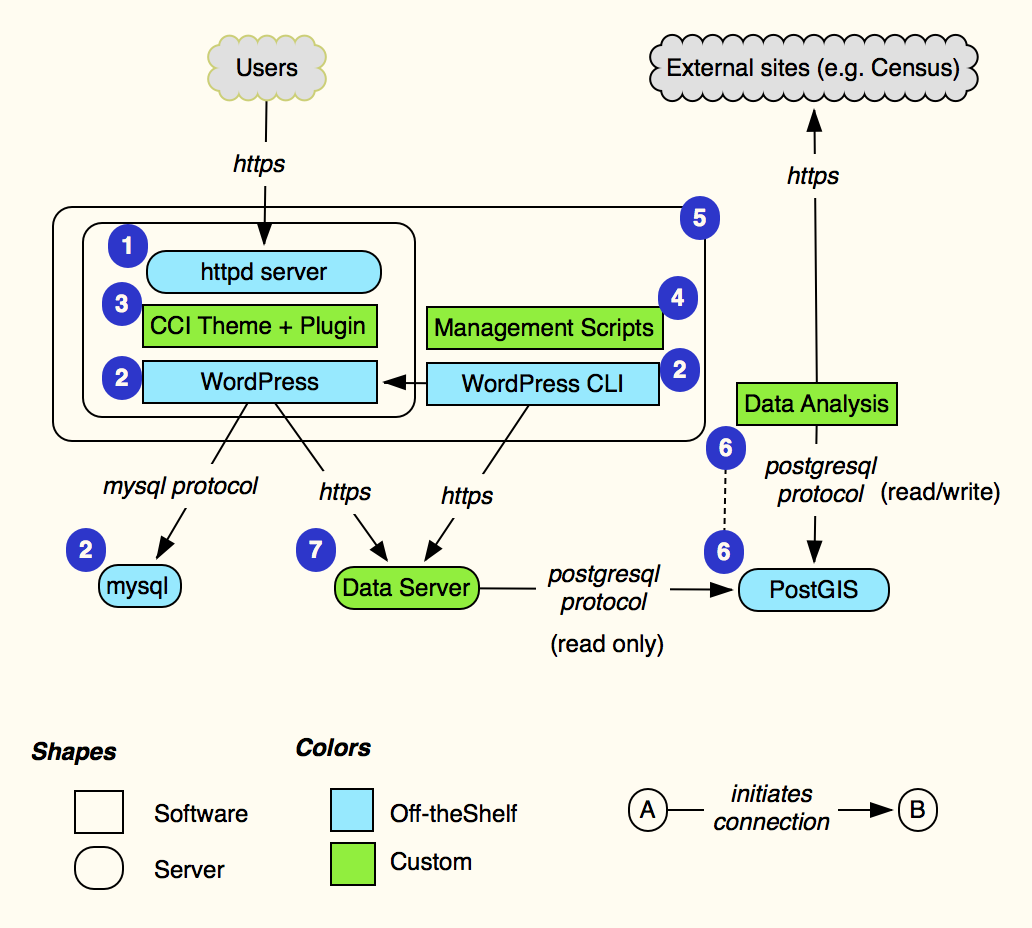
1. WordPress (4.7.3): BCI uses it as a tool for Content Management System.
2. MySQL (5.5): WordPress uses it as its backend data storage.
3. Red Hat Apache (2.4): BCI uses it as its web server.
4. PostGIS (2.3.4): BCI uses it to manage Graphical Information System (GIS) data.
5. PostgreSQL (9.6): PostGIS uses it as its data store.
6. System components

The current BCI system consists of the following components:

1. Apache Web Server
2. WordPress with the Multisite Features enabled
3. WordPress Plugin:
4. Import/Export plugin
5. Member plugin
6. WordPress Command Line Interface (CLI)
7. WordPress CLI Management Scripts
8. BCI (CCI) Tool:
9. Theme
10. Plugin
11. PostGIS Database
12. Data Analysis Component
13. Data Server API

### **3.3 System Architecture**

The system architecture can be illustrated in the following diagram:



The components in the diagram are described below:

1. **Web Server**: Red Hat Apache 2.4. web server, configured to serve a WordPress Installation. All external user interactions pass through the webserver to be handled by WordPress components.
2. **WordPress**: BCI uses it to deliver its dynamic content through Internet and manage its web sites. WordPress works with an Apache web server to manage its web interface. WordPress uses MySQL to store all its data, including all the data users generate. BCI uses the standard WordPress installation with “MultiSite” features enabled. No change is made to the WordPress code base.

**WordPress Plugins**:

1. Import/Export plugin: used to import content
2. Member plugin: used only for testing.

**WordPress Command Line Interface** (CLI): is a utility that consists of a collection of executable, scriptable commands for building programs that populate BCI Sites and administer a WordPress installation. Currently it is restricted to running on the same server as the WordPress installation.

1. **BCI (CCI) Tool (Theme & Plugin)**: built as custom plugins to WordPress, which consists of the following two components:
2. Theme: consist of a collection of PHP (Red Hat PHP 5.6) code, CSS and Ajax to provide “look & feel” for WordPress installation.
3. Plugin: a package of PHP code used to customize a WordPress installation, to configure and enforce User roles, manage custom post types (like Prompts, and Responses), and configure administrative interfaces so that Team members will have an effective, though bare-bones, management environment.

The BCI Tool is structured as a WordPress Web Application. In essence, the BCI Plugin and BCI Theme that implements the BCI Tool leverage core components of WordPress, while providing an alternative interface to the primary end. The WordPress administrative interface, however, is selectively available to NTIA staff (for content management) as well as to Team Leads for user management. As noted, the BCI tool provides the ability to create and remove users, assign user privilege levels and manage user passwords. These capabilities, aside from user privilege levels, are based on those provided in WordPress.

The BCI Plugin and Theme follow the normal WordPress structure for plugins and themes: the plugin provides core functionality, while the theme provides the most of the look and feel.

This distinction currently gets muddled in several places but can be resolved as the code matures. For example

* The plugin often wraps content in HTML div elements for later styling. In this case, the style identified in the div is defined in the theme. Such usage is documented in the themes style.css file.
* The order of sub elements on Page is occasionally defined in the Plugin.
* Some Theme pages (primarily the top-level pages) embed content.

1. **WordPress CLI Management Scripts**: custom scripts written in Linux shell (bash) with calls to WordPress CLI and BCI Data Server. The scripts is used to spin up a new Site within the BCI installation, which also allow access to the BCI Data Server in order populate the content for a BCI Site.
2. **The NTIA server** minimally needs to host the Web Server, the WordPress Installation with plugins, the WordPress CLI utility, BCI Tool, and WordPress CLI Management Scripts.
3. **PostGIS Database**: composed of the PostgreSQL database server with extensions and tools for managing Graphical Information Systems (GIS) data. It is loaded with Census shapefile data and FCC tabular data (subscription and deployment). Currently the NTIA CPS data is managed as a single file on the server where the Data Analysis Component runs.

**Data Analysis Component**: composed of a small amount of Perl code and a pair of R (3.3.1) packages that use GIS from the PostGIS database with data drawn from FCC, NTIA and Census to generate table and images. The component reads and writes data to Post GIS. BCI Tool presents the output of the component to the users.

R is used with the following libraries:

* + testthat
  + curl
  + DBI & RPostgreSQL
  + ggplot2, ggmap
  + mapproj
  + gpclib
  + rgeos
  + scales
  + plyr
  + assertthat
  + jsonlite
  + raster
  + others….

R libraries can be installed at a user-level, but some system-level support libraries are also needed. Here are a few of the key ones; and like all system libraries, they need several more.

Linux/C libraries to support R:

* gdal
* geos
* proj4
* others….

1. **Data Server API**: implemented in PERL-based framework “Mojolicious”; used to mediate between the Data Analysis Component and the WordPress sites. It is introduced in order to separate and serve the data views generated by the Data analysis Component from the rest of the system. The server provides a light-weight implementation of an HTTP-based API.

The Data Server API can be used in either of two modes:

1. If the Data Server API can be exposed to end users, then images, such as maps, stored in the Postgres server are retrieved by direct access. This is the mode used in testing in the Cloud.
2. When the Data Server API has to be hidden from users, the images and tables for a given locality are pre-loaded into the WordPress Media Library. This takes a small amount of time when the site is created, or when the images need to be updated. The software can be configured manage either images or tables via configuration parameters. There are two parameters; one for tables and the other for images. See Appendix II: . When both are set to use the media library, the settings for the Data API are ignored, and the data server itself does not need to be running. Appendix II: documents the settings and their usage.

The Data Server API is written in PERL using the DBD::Pg, JSON::PP, and Mojolicious::Lite packages. On the NTIA server, Red Hat PERL 5.24 with DBD::Pg and JSON:PP is installed from Red Hat modules, and Mojolicious is installed via CPAN.

### **3.4 Website Navigation**

The following diagram describes the navigation of the BCI website.



The Define my Team Tab in the diagram contains 1 module while the Assessment, Adoption, and Community Tabs contain 4 modules each (see below for module definitions).

# Modules Defined in Version 1

|  |  |  |
| --- | --- | --- |
| Aspect | Module | Internal Identifier |
| Community |  | community |
|  | Leadership | leadership |
|  | Community Priorities | community\_priorities |
|  | Policy and Environment | policy\_and\_environment |
|  | Stakeholders | stakeholders |
| Access |  | access |
|  | Broadband Access | broadband\_access |
|  | Mobile Access | mobile\_access |
|  | Provider Engagement | provider\_engagement |
|  | Public Assets | public\_assets |
| Adoption |  | adoption |
|  | Adoption and Use | Adoption\_and\_use |
|  | Digital Inclusio | digital\_inclusion |
|  | Digital Skills | digital\_skills |
|  | Devices | devices |
| Define my Team |  | get\_started |
|  |  |  |

WordPress Pages are basic elements used represent a single page within a blog/workbook. Pages have an associated ‘page template’, which indicates a template within the Theme that should be used to display the contents of the Page.

The CCI Tool leveraged the notion of Pages to define the CCI Modules. Modules in CCI are not defined by custom post types.

1. Each module is defined by an element in the Module category, as  
   defined above.
2. Each Prompt in a Module has the module category, and an order (as specified by the menu\_order field. Thus the sequence of Prompts in a module is easily determined.
3. The Page that displays the sequence of Prompts is nothing more than an ordinary WordPress page that has three attributes:
   1. The module category
   2. It uses as a template the ‘Module Assessment’ template defined in the Theme.
   3. It has a slug (internal name) of the form <module\_name>\_assessments;
4. The Page that displays the sequence of Responses is nothing more than  
   an ordinary WordPress page that has 3 attributes:
   1. The module category
   2. It uses as a template the ‘Module Responses’ template defined  
      in the Theme.
   3. It has a slug (internal name) of the form “<module\_name>\_responses”. Code in the plugin depends on this naming convention to find the Response page for module.

Given this convention, each Module has two associated, statically-define pages that are used for displaying the Prompts and their associated Responses (including the Team Responses).

The Admin pages can only be accessed internally. The WordPress users can access content for editing using the Admin pages. Access to the pages is limited within the BCI Tool. However, since there is a new Admin page for each custom post type, the BCI Plugin provides customization to various Admin pages in order to better support end users.

### **3.5 Network Considerations**

BCI exposes ports 80 and/or 443 on the web server via the HTTP daemon. HTTP rewrites rules that can be used to moderate administrative access. WordPress also creates its own rewrite rules (for managing links and multi-site), so care will need to be taken.

The Data Server API in its ‘hidden’ configuration can run on localhost, port 8080. This can be changed in the cc-config.php file.

MySQL and PostGIS do not require external ports unless they are run on separate servers.

### **3.6 Deployment Considerations**

The deployment options are tested in both hardware configurations (NTIA Server, cloud.gov) mentioned above.

The NTIA server minimally needs to host the Web Server, the WordPress Installation with plugins, the WordPress CLI code, and the management scripts. All other servers can run either locally or remotely depending on deployment decisions.

The Data Analysis Component can be installed on any server that has access to the PostGIS database. As a deployment option, the Data Server API code can be combined with the Data Analysis Component as one deployable component.

**Startup**

* Httpd is started by the httpd24 service
* MySQL is started by the mysql service
* PostGIS is started by the postgres services on the server
* The Data Server API should be started by an ordinary redhat service (work in progress), but the init scripts need to be written.
* The Data Analysis Component is triggered by a script, run by an ordinary user having access to the PostGIS database.

# Database Design

In the BCI system, all custom tables are created in its PostgreSQL database, which consist of three groups:

1. TIGER data tables as managed by PostGIS
2. FCC data tables, for which the raw data is provided by the FCC in the form of CSV files
3. CCI Result tables which store computed results that are derived from the Data Analysis component.

In general, the Data Analysis component code reads from the TIGER and FCC tables, and both read and write to the CCI result tables (3).

### **4.1 TIGER Tables**

The BCI Tool has benefited from the open data sets and API’s supported by the United States Census Bureau, and also from the open source projects PostgreSQL and PostGIS. The TIGER tables are document at <https://www.census.gov/geo/maps-data/data/tiger.html>

Version 1 of the data analysis tools imports a series of national level tables from the CENSUS using a script pg\_import\_nation.sh, which was generated by the PostGIS loader\_generate\_nation\_script as documented at <http://postgis.net/docs/Loader_Generate_Nation_Script.html>.

We then use a heavily edited version of the loader\_generate\_census\_script (<http://postgis.net/docs/Loader_Generate_Census_Script.html>) to import state-level tables for tracts, blocks groups, blocks, places, and county subdivisions (cousub). The edited version of the script is called pg\_import\_state.sh.

Both scripts are usually run as the postgreSQL user.

### **4. 2 FCC Tables and other Data Tables**

The following tables reside in the cci\_tables schema. These tables are loaded from data files, and updated infrequently.

The primary use for this collection of tables is to provide input to the data analysis component.

| **Table Name** | **Use** |
| --- | --- |
| **cci\_tables.fcc\_477** | **FCC Fixed-line data** |
| **cci\_tables.fcc\_477\_troves** | **Meta data about the FCC fixed-line data** |
| **cci\_tables.fcc\_477\_mobile** | **FCC mobile data** |
| **cci\_tables.fcc\_477\_mobile\_troves** | **Meta data about the FCC mobile data** |
| **cci\_tables.fcc\_quintiles\_jun\_2015** | **FCC adoption data** |
| **cci\_tables.fcc\_quintiles\_dec\_2015** | **FCC adoption data** |
| **cci\_tables.fcc\_quintiles\_troves** | **Meta data about the adoption data** |
| **cci\_tables.techcodes** | **Technology code** |
| **cci\_tables.cci\_provenance** | **Table summarizing the sources for data item** |
| **cci\_tables.USPS\_cousub\_500k** | **One table for each state, with county subdivision cartographic boundaries. Example: cci\_tables.mi\_cousub\_500k for Michigan.** |
| **cci\_tables.USPS\_place\_500k** | **One table for each state, with place cartographic boundaries. Example: cci\_tables.mi\_cousub\_500k for Michigan.** |
| **cci\_tables. us\_cb\_county\_500k** | **Table of cartographic boundaries for all US counties** |

### **4.3 CCI Result and Support Tables**

The following tables reside in the public schema. In general, the data analysis component manages the content of these tables, while the Data Server API reads from the tables to provide data elements.

When the system is configured to serve tables and images from the WordPress media library, only the cci\_marker table is pertinent.

| **Table Name** | **Use** |
| --- | --- |
| **cci\_marker** | **Marks whether the data elements for a geoid have been computed.** |
| **cci\_image** | **Stores images (maps) for a geoid** |
| **cci\_json** | **Stores json-encoded tables for a geoid** |
| **cci\_api\_authkey** | **Stores valid authorization keys** |

# System Security

The BCI uses SSL with server-side certificates for authentication to ensure connection through the Internet is encrypted. The user authentication and authorization of the system are managed by the built-in security features of WordPress.

### **5.1 Role-Based Access Control**

The User-level access to the BCI system is moderated via the WordPress mechanisms. Users are created as all WordPress users, and their profile information including login credentials are managed by the WordPress platform.

However, the actual user roles exposed by the BCI Tool are not those of default WordPress. Instead, the prototype provides roles that model the actual end-use requirements for participating an assessment or managing the content of the assessments. ***Appendix V*** provides a cross-reference between the roles and the capabilities, as implemented in the BCI Plugin.

The user roles have been reconfigured as follows:

* 1. The ‘standard’ WordPress roles of ‘Author’, ‘Editor’, and ‘Subscriber’ are removed by the BCI plugin. These roles can no longer be assigned. Since removing roles is a one-time operation, these roles will not reappear in a WordPress installation if the Plugin is deactivated or deleted.
  2. The BCI Plugin adds and enforces appropriate access for the following roles:

Roles for External (non-NTIA) Users:

* **Community Participant**: Can only read published Responses and Team Responses. Commenting is envisioned, but not yet implemented.
* **Community Partner**: The overarching design envisions a Community Partner as someone able to view several Workbooks simultaneously and in aggregate. In Version 1, this role is limited to being the same as a Community Participant having access to more than one Workbook.
* **Contributor**: Can edit and publish their own Responses, plus the capabilities of a Community Participant.
* **Team Editor**: Can and edit Team Responses, plus the capabilities of Contributor.
* **Team Lead**: Depending on configuration, can invite other persons to join the team, assign their roles, plus the capabilities of a Team Editor. A configuration setting makes it possible to restrict Team Leads to having the same capabilities as a Team Editor, so that Team Leads would not be able to make changes affecting the users on their Team.

External users are not permitted to add, edit, or remove Pages, Posts, Prompts, Modules, or Resources. Nor are they permitted to change the Site configuration settings including the Theme and installed Plugins.

Roles for NTIA (staff) Users:

* **Assessment Editor**: Can add, remove, edit, and export Prompts, Resources, Pages, and Report Elements.
* **Assessment Administrator**: Can access multiple sites, and view aggregated data across sites. This role is not implemented in Version 1.
* **Administrator** (within a site) **or Network Administrator** (across sites): Retains the full access rights as in WordPress, plus the ability to add, edit or delete the data elements introduced by the BCI Plugin.

### **5.2 Component Access**

• The WordPress and HTTP components run as the “apache” user on the web server.

• The data API runs as the “cciuser” user.

• The MySQL server runs as the “mysql” daemon user on the server. WordPress accesses the database within the server using an ordinary mysql user configured according to WordPress requirements.

• The PostGIS server runs as the “postgres” daemon user on the server. Access to BCI data is provided to an ordinary PostGIS user. This user needs SELECT, INSERT, UPDATE, DELETE, and TRUNCATE privileges from the data analysis component, but only SELECT privileges when accessed from the data API. This separation can be enforced by creating two different roles in PostGIS.

### **5.3 Auditing**

The BCI system itself does not provide any auditing capability within the system. It relies on the network facility to perform the auditing functionality.

# Appendix I. Terminology

**NTIA Terms**

**Account**: A relationship between the NTIA Technical Assistance team and one or more people in a locality.

**Locality**: Any place where broadband planning is occurring.

**BCI Tool Term**

**Site**: A single instance of an Assessment. A Site is limited to a single locale. Operationally, a site is one of several “sites” in a WordPress Multisite configuration. Creation of a new Site/Account is managed by a script run by an administrator on the server. The script is written in bash and uses the WordPress CLI facilities and the data server, to create the site.

**Locale:** A single geographical region defined by the US Census that is either a U.S. County or a Census Defined Place. Version 1 of the tool operates only on these two types of localities. Every locale is uniquely specified by a geographical identifier specified by the US Census Bureau. For example, ‘23’ is the geoid for the state of Maine, ‘23029’ is the geoid for Washington County in Maine, and ‘2302795’ is the geoid for the city of Bangor, Maine.

**Team:** One or more users, defined to the BCI Tool, who are collaborating together to complete or use an Assessment. Each team has one or more designated Team Leads, who control the membership in a Team. Teams are specific to workbooks. While a single user can be a member of more than one Team (and therefore access more than one site), Version 1 of the tool does not share data across workbooks, even to users with administrative privileges.

**Workbook:** A Workbook contains the content and Responses for a single Team working on planning for a single locality. Operationally, a workbook is represented by a single “blog” in a WordPress Multisite configuration. Creation of a new Workbook is managed by a script run by an administrator on the server. The script is written in ‘bash’ and uses the WordPress CLI facilities and the data server, to create the workbook. Workbooks are structured as a collection of modules that, taken whole or in part, provide the basis for a community’s assessment effort.

**Module:** An ordered collection of Prompts that can include text, tables, or graphics.

**Prompt:** A single question or comment within a module. *Actionable* Prompts pose questions that can be answered.

**Response:** A single answer to an actionable Prompt. A given user can create only one Response to each Prompt. but each contributing member of a team can provide an alternative Response.

**Team Prompt:** Each team can provide at most one final Response to each Prompt. Final Responses become part of the Report.

**Report:** Output from an Assessment.

**Report Element (or ‘Element’ for short):** A sub-part of a Report.

**Resource:** A citation, link, or other description of materials that may be useful to the team in continuing their planning efforts. In the tool, resources are coupled actual Responses.

**WordPress Terms**

**WordPress**:A content management system, originally used for weblogs, but increasingly used to deliver web-based applications. WordPress requires two external components: a web server to manage the Web interface, and a MySQL database to manage the dynamic content. WordPress is also scriptable from the command line using the “WordPress CLI tool.”

**WordPress Command Line Interface (CLI)**:A collection of executable, scriptable commands that can be used to build programs that administer a WordPress installation. While the WordPress CLI can be used over a network connection, its use in BCI is so-far restricted to running on the same server as the WordPress installation.

**Theme**: A packaged collection of PHP code, styles, fonts, and AJAX/Javascript components used with a Plugin to give a WordPress installation its “look and feel”.

**Plugin**: A packaged of PHP code used with a Theme to customize a WordPress installation.

**Blog**: A WordPress instance of a web log, where content lives. In the BCI Tool, we replace the notion and style of a blog with a Workbook, but it helps to be able to use the “blog” word when working with WordPress details.

**Multisite**:WordPress provides a mechanism to manage multiple as a single administrative group and single WordPress installation. We are using these features to support isolation between various BCI Sites while easing the administrative overhead. For example, in a Multisite deployment, there might be 20 Workbooks (blogs), but only a single installation of the BCI Theme and Plugin.

**WordPress Media Library**:An area of disk storage, on the WordPress server, were images and other data can be stored and shared via the WordPress front end.

# Appendix II: Custom Taxonomies

WordPress Custom Taxonomies provide ways to build tagging schemes for data elements. In WordPress, a taxonomy can be defined as hierarchical, and can be treated either as categories (only a single value is allowed at each time) or as tags (an object can have multiple tags).

The CCI Tool introduces three custom categories, each implemented by a custom taxonomy. Each of categories is assumed to be hierarchical with only a single value selected. The values are built-in and should not be user editable.

**CCI Categories defined using Custom Taxonomies.**

*Module*

A taxonomy for the CCI Modules. This taxonomy is omni-present in the code: module pages use this value to define the Module; every Prompt is in a single Module, and each Response to a Prompt is tagged with the module tag of the Prompt. This category appears in the code as cci\_module\_category. The Module elements are set during plugin installation, and are defined in the code. The list of modules and their tags appears in the section entitled *“Modules Defined in Version 1*”

*Response Type*

Defines the expected response-type for a Prompt, such as yes/no or checkbox. Each response type has its own PHP templates for formatting and its own handler for managing responses. This category appears in the code as cci\_prompt\_category. The Response Type values are set during plugin installation, and are defined in the code. The list of response types and their values appears in the section entitled *“****Error! Reference source not found.***”

*Source*

Defines a small set of possible sources for Resources. This makes it possible to sort on the resource types. Each Resource has a single source. New source tags can be added as the Resource descriptions are defined.

**CCI Tags defined using Custom Taxonomies**

The CCI Plugin creates a single tagging scheme that is used to tie Resources back to Responses.

*Resource Link Tag*

A tag used to connect outcomes of questions with resources. In general, this tag ties together a many-to-many relationship among Prompt answers and resources.

This scheme is not as user-friendly as it might be, but we’ll leave that for the next major code release. The basic idea is that each actionable Prompt has one or more outcomes: for instance ‘Yes’ or ‘No’. The current implementation allows an Assessment Editor to associate a single “resource link tag” with each answer. For example, the “learn more about municipal networks” tag might be associated with the ‘Yes’ outcome. As resources are added, several of the Resources might also be tagged with “learn more about municipal networks”.

If the Team’s response is ‘Yes’ to this Prompt, then, all of the resources tagged with “learn more about municipal networks” will appear in their Report.

Two notes:

1. This scheme allows the Assessment Editors to add and remove Resources independently from assigning recommendation tags to Prompt outcomes. You can add a new resource at almost any time.

2. The implementation usually allows an Assessment Editor to assign a resource link tag to the converse of each outcome. This is overkill for ‘yes/no’, but quite nice for multiple-choice questions since each choice can have a tag for ‘checked’ and ‘not checked’.

The resource link tag currently appears in the taxonomy declarations as cci\_recommendations\_tag.

# Appendix III: Custom Post Types and Custom Post Metadata

**Custom Post Types (CPTs)** are primary extension mechanism in WordPress for adding, displaying, and editing new forms of content. In the CCI Tool, Prompts, Responses, Resources, and Report Elements are all implemented as CPTs.

**Custom Post Metadata** consists of data fields associated with a given post. The custom post types created by the CCI Plugin relies heavily on post metadata to store additional information unique to the post type. For example, the actual user-supplied answers to a Prompt are stored using metadata fields tagged to the Response

The linkage between posts and post meta data items is maintained by WordPress, including during post deletion, export, and import.

## The Custom Post Types

Three custom post-types are defined in the CCI Tool:

**Prompt**: Represents an assessment question. Prompts can be children of other Prompts

**Response**: represents a single answer to a Prompt.

**Resource** represents any of the Resources associated with a specific answer to a Prompt. Much of the code base uses the term ‘Recommendation’ synonymously with ‘Resource’.

**Report Element (Element)**: represents a single chunk of a report.

## The Prompt CPT

Prompts define a question with a set of values, and descriptions to go with the values.

* All values are encoded as strings.
* Values and descriptions are stored as an array of arrays, (answer, description)
* When the response type has a variable number of values (e.g., radio or checkbox, the values and descriptions are stored in the Prompt metadata.
* Each Prompt choice has an associated outcome.
* Each outcome can have a single recommendation tag and a single anti recommendation tag. These tags are the “resource link tags” discussed above.
* Use parent/child relationships for follow-on questions.

The key data attributes for a Prompt include:

|  |  |
| --- | --- |
| Key Attributes of a Prompt | |
| Name | **Purpose** |
| Response\_type | from the response\_type category |
| Module | from the module taxonomy |
| Title | Abbreviated title |
| Content | Text of the Prompt. See  *Prompt* Content*.* |
| Outcome values | Defined by the Response Type, and encoded for each Prompt using post metadata |
| Order in module | Uses the menu\_order field |

For follow-on questions, three additional fields become active. In general, the follow-on Prompt is activated when “parent-value op my-value” is true. So for a follow-on Prompt, we need both the op and the parent-value to compare against.

|  |  |
| --- | --- |
| Additional Metadata for Follow-on Prompts | |
| Name | **Purpose** |
| Parent | The preceding question |
| Operator | The comparison operation to be used |
| Value of Parent | The value of the parent to be compared |
| Prompt Values | Type, with additional interactions provide by the additional meta-data fields described above. |

### 

### Prompt Content

The content field of a Prompt is used to provide both the “titling” text of the Prompt as well as any additional information. We moved to this concept when we realized that the text of many Prompts is longer than the text allowed by WordPress’s default title field. The content of Prompt is structured as a series of blocks (implemented by HTML div elements), with the first div having the style ‘prompt-question’. Subsequent Prompts are styled as ‘prompt-more’. This transformation was implemented during bulk-loading of the content. During the bulk loading, titles were then shortened to 64 characters, since they appear only during editing.

### Shortcodes for Data Items

**Shortcodes** are a WordPress mechanism that allows a content editor to create content that, when displayed, triggers code execution. As an example, the data component interface consists of a set of short-code definitions that simplify editing while hiding the complexity of fetching a data table and rendering it into HTML.

All of the shortcodes use a global setting for the Workbook’s geo-identifier.

|  |  |  |  |
| --- | --- | --- | --- |
| Table of Short Codes for Data Access | | | |
| Name | **Usage** | **Example** | **Options** |
| locale | Name of the locale | [locale] |  |
| state\_name | Long name of the state | [state\_name] |  |
| acs\_demographics | ACS Demographic table | [acs\_demographics] | rows |
| acs\_internet | ACS internet table | [acs\_internet] | rows |
| cps\_internet | CPS internet table | [cps\_internet] | rows |
| topline\_access | Topline access numbers | [topline\_access] | rows,cols |
| topline\_adoption | Topline adoption numbers | [topline\_adoption] | rows,cols |
| cci\_image | Creates an img tag | [cci\_image name='work\_from\_home'] | name,state,height,width |
|  | Image for a state area | [cci\_image state='y' name='work\_from\_home'] |  |
| cci\_table | Full table of data | [cci\_table name='work\_from\_home'] | name,state,rows,cols,labels |
|  | Table for a state area | [cci\_image state='y' name='work\_from\_home'] |  |

The various tabular shortcodes use the following options. In all cases the values of the option should be a quoted string. Multiple values for a single option are comma-separated.

|  |  |  |  |
| --- | --- | --- | --- |
| Options for ‘cci\_table’ | | | |
| Name | **Usage** |  | **Format** |
| name | required | The name of the table | name='<value>' |
| state | optional | Set to’y’ for entire state data | state='y |
| rows | optional | Indices of rows to select | rows='<value>,...' |
| cols | optional | Names of columns to select | cols='<name>,...' |
| labels | optional | Labels of selected columns | cols='<name>,...' |

The cci\_image shortcode accepts the following options. In all cases the values of the option should be a quoted string.

|  |  |  |  |
| --- | --- | --- | --- |
| Options for `cci\_image` | | | |
| Name |  | **Usage** | **Format** |
| name | required | The name of the table | name='<value>' |
| state | optional | Set to ’y’ for entire state data | state='y’ |
| height | optional | Height of image | height='number’ |
| width | optional | Width of image | width='number’ |

### Prompt Navigation

In Version 1 there can be only one single follow-on question to each Prompt outcome. If the user responds ‘Yes’, the Assessment Editor can code a follow-on question, but only one. This does not appear to be a major limitation, since the Version 1 content does not use any follow-on questions at all!

Prompts within a module are sequenced by the menu\_order field, and have a post\_parent field == 0. Follow-on questions have the post’s parent field set to the ID of their originating Prompt.

The editing interface (admin page) for Prompts allow the Assessment Editors to manage follow-on questions. Version 1 implemented the approach of specifying the follow-own in the actual follow-on question. One can model this user interaction as the Assessment Editor saying “Display this Prompt if the answer to some other Prompt is X.”

In the present implementation, you can provide more than one follow-on question, and only one of them will be selected to be presented. This is a limitation in the implementation.

### Post Data and Metadata used for Prompts

|  |  |
| --- | --- |
| WordPress representation of a Prompt | |
| Name | Implementation |
| response\_type | from the prompt\_type category |
| module | from the module taxonomy |
| title | Abbreviated title |
| content | Text of the prompt. See “  Prompt Content” |
| Outcome values | Defined by the Response Type, and encoded for each Prompt as post metadata |
| position | Uses the menu\_order field for now |
| unique id | Uses the ‘slugname’ of the post to provide a unique name to each prompt. |

### Representing Prompt Answers in WordPress Post Metadata

The actual answers attributes are implemented as follows. Here, ‘N’ is an index in the range 0..number of answers.

|  |  |  |
| --- | --- | --- |
| Answers (Outcomes) to Prompts | | |
| ID | Tag | Usage |
| prompt\_id | \_answer\_N | Indicates that the Nth outcome is present |
| prompt\_id | \_value\_N | string that describes the value |
| prompt\_id | \_recommendation\_N | Resource link associates with value\_N |
| prompt\_id | \_anti\_recommendation\_N | Anti-recommendation associated with value\_N |
| prompt\_id | \_comparison\_op | Follow-on question comparison operator |
| prompt\_id | \_comparison\_value | Follow-on question comparison value |

## The Response Custom Post Type

A Response represents a single answer. Each individual user can create a single Response to an actionable Prompt. In addition, each team can formulate a single Team Response to a Prompt.

Responses are the only way that external users can introduce data into the CCI tool database. In general, a Response encodes the Prompt that it is answering, the author of the Response, a post status that could be a Team status, and the actual outcome values. In addition, from the values of the outcomes, and the resource link tags in the Prompt, it is straightforward to compute a list of associated resources.

The plugin actually defines a new ‘post status’ value to represent Team Responses.

### Wordpress Data attributes for a Response.

The key attributes for a Response include:

* The Prompt for this post (metadata)
* Author
* Actual of value or values for the Response
* Status of the response: typically ‘Published or Team’.

These attributes are implemented in Wordpress as follows:

|  |  |
| --- | --- |
| Attributes for Responses | |
| Name |  |
| prompt\_id | The Prompt for this post as post metadata |
| author | The built-in author field of the post |
| post\_status | The built-in post\_status field of the post |

### Response Post Metadata Usage

The Response values are implemented as follows. As with Prompts, ‘N’ is an index in the range 0..number of answers to Prompt.

|  |  |  |
| --- | --- | --- |
| Representing Response Answers | | |
| ID | Tag | Usage |
| response\_id | prompt\_id | Id of the originating Prompt |
| response\_id \_ | answer\_N\_ | value\_N\_ – drawn from the prompt\_id’s data |
| response\_id | \_descr\_N\_ | string that describes with value\_N drawn from the prompt\_id’s data |
| response\_id | \_recommendation\_ | string that describes the recommendation drawn from the prompt\_id’s data |
| response\_id | \_anti\_recommendation\_ | string that describes the anti\_recommendation drawn from the prompt\_id’s data |
|  |  |  |

## The Resource (Recommendation) Custom Post Type

This post type represents a single Resource. The key data attributes include

|  |  |
| --- | --- |
| Key attributes of Resources | |
| Content | The text of the resource, possibly including URLS |
| Weight | One of High/Medium/Low |
| Source | A source tag |
| resource link tags | One or more resource link tags shared with prompt values |

### Resource Post Metadata Usage

The Response attributes are implemented in WordPress as follows

|  |  |  |
| --- | --- | --- |
| Response Attributes as Wordpress Metadata | | |
| ID | **Tag** | **Usage** |
| recommendation\_id | \_reco\_weight\_ | High/medium/low |

## The Report Element CPT

The report element is a very thin veneer over the normal post type. Report Elements only appear within a final report. It is pretty much modeled as a normal post with page attributes, and a module identifier.

|  |  |
| --- | --- |
| Key Attributes of an Element | |
| Name | **Purpose** |
| Module | from the module taxonomy; this can be module or a one of the parent categories, like ‘Access’. |
| Title | Abbreviated title |
| Content | Text of the Prompt. See  *Prompt* Content*.* |
| Order in report | Uses the menu\_order field WordPress attribute |

The theme page for the Report Template emits the report elements in order.

As with access to data items, a suite of short-codes is provided that will allow an Assessment Editor to embed specific prompts and responses into a Report Element. Two short codes have been provided, corresponding to “recall the text of a prompt” and “recall the team’s answer to a prompt. In addition, there are two options to the rendering of responses: an option to display scaled values as images (like ) rather than as numeric values, and an option to display the text of the prompt

|  |  |  |  |
| --- | --- | --- | --- |
| Table of Short Codes for Data Access | | | |
| Name | **Usage** | **Example** | **Options** |
| prompt | Text of a prompt | [prompt id=’q\_138’] | id |
| response | Value of the team response | [prompt id=’q\_138’] | id, image,prompt |

The ‘id’ parameter should always be the unique id of a Prompt. The ‘image’ and ‘prompt’ options should be either ‘y’ for yes or ‘n’ for no; both default to ‘n’ (no).

# Appendix IV: Configuration Options Defined in Version 1

The following options can be defined in the file ‘cci-config.php’.

|  |  |
| --- | --- |
| CCI\_API\_URL | Fully-qualified URL to the data server. This option can be left undefined if the Media file options are both set. |
| CCI\_API\_KEY | API key for the data server. |
| CCI\_IMAGES\_FROM\_MEDIA\_LIBRARY | Define this constant to be true to serve images directly from the WordPress media library. Using this option requires that all images be loaded into the Media Library directly, instead of using the data server. |
| CCI\_TABLES\_FROM\_MEDIA\_LIBRARY | Define this constant to be true to serve data tables directly from the media library. Using this option requires that all images be loaded into the Media Library directly, instead of using the data server. |
| CCI\_TEAM\_LEADS\_MANAGE\_USERS | Define this constant to be true to allow team leads to manage their own team’s users. |
| CCI\_BBUSA\_LINK | Preferred external link to BBUSA for use in branding. |

# Appendix V: BCI User Roles and Privileges

WordPress uses Roles and Privileges/Capabilities to define the actions that is allowed to take, and gather sets of actions into a coherent whole. For example, the Team Contributor role is fully defined by a set of capabilities many of which are defined by WordPress, along with some new capabilities defined by the BCI Plugin. For example, the capability “edit\_prompts” was added to handle controls over which roles can edit the newly defined Prompt custom post type.

This appendix shows the privileges/capabilities that each user role has in terms of WordPress.

