FHIR Spreadsheet Authoring

This page describes the process for making use of Excel (or OpenOffice) spreadsheets to design and maintain FHIR data types, resources and profiles. If you haven't already, please read FHIR Guide to Authoring Resources or FHIR_Spreadsheet_Profile_Authoring for important contextual information on how to set up the build environment to make use of the spreadsheet you're about to edit, where to find the appropriate template, etc.

The use of spreadsheets to develop and maintain profiles is expected to be a near term solution with more sophisticated (and user-friendly) tooling currently under development. There are no plans to invest in similar enhanced tooling for authoring resources because there aren't going to be that many and they'll all be developed within HL7.

A single set of documentation has been provided because most of the tabs and columns are identical regardless of what type of artifact is being created or maintained. Where the rules don't apply everywhere, this page will flag the rule as **Resource/Data Type-only** or **Profile-only**.

Contents

Excel Spreadsheet

The Excel spreadsheet contains the main logical definitions of the resource, data type or profile. In order to support version control and other forms of text processing, the spreadsheet is stored as an XML document. In Excel, this format is chosen by saving as an XML Spreadsheet 2003. Any other software that can edit this format (e.g. OpenOffice) can also be used. It's even possible to use a text editor, though this is not recommended. (Note: the project team is not committed to Excel. Alternatives can be considered, as long as the same logical content can be defined).

Some general notes on the formatting of the spreadsheet:

- Lookup tables and validation formatting only exists inside the solid black border. Rows should not be added outside of this border. Columns may be added outside the border (and are less likely to end up with formatting problems if added there)
- Fields where data can be entered are colored white. The font is generally black. A dark-blue font indicates that markdown text (with hyperlinks, formatting, etc.) can be present.
- Fields which are dark-grey should not have content entered. The whole row will turn dark-grey if you put a "!" in front of the value in the "key" column (the first column except for profile structures tabs where it's the "Element" column). This indicates that the row is considered to be commented out.
- Fields which are light-grey indicate that the cardinality for the row has been set to 0..0. Values may still be entered (e.g. the 'requirements' cell might be populated to explain why the element was disallowed), however in most cases, there will be little point in doing so
- Fields which are red indicate that a validation check has failed. Consult this documentation for guidance as to what the problem might be. If you think a field has been highlighted in error, alert one of the FHIR tooling team members (probably via the committer Skype chat.)
- Fields which are orange indicate that a validation check has failed, but that the WG doesn't necessarily have to fix it - v2 and v3 mappings can be handled by the FHIR core team if requested.
- Fields which are yellow indicate a warning. This is generally where a field is empty that is considered to be a "SHOULD". It's ok to leave it blank, but you should have a good reason for doing so.

Additional notes:

- Many of the cells in the spreadsheet have lookups or validation rules or both. Excel makes it unpleasantly easy to mess these up. Guidance is as follows:
    - Cutting, copying and pasting cells can all do bad things. In general, inserting rows in the middle is fine. At the top or bottom can be problematic. Cutting anything will tend to cause grief.
    - If you want to paste, try to paste just values so as not to replace the validation and look-ups.
    - If you want to add a new column to capture some additional piece of information, copy the "ToDo" column and then paste it where you want it - that way you won't end up with validation rules you don't want.
- If your formatting gets totally messed up, ask Lloyd for help and he should be able to re-apply the template to your existing content, fixing all the formatting and lookups.
- The only cells that have validation rules and look-ups are those inside the solid background. If you find you need more rows than the set provided, ask a question on the FHIR committers list as there may be something wonky with your design. (There should be more than enough rows for most FHIR purposes.)
- There are comments explaining the use of each column in the header for that column (If you hover the mouse over, you'll see a hint). If you need further documentation, you should find it here on the wiki. (Note: because we save our spreadsheets as XML, there's a limit to how much text we can include in a comment and actually have it displayed.)

- Where drop-downs exist, that doesn't necessarily mean that you're limited to the contents of the drop-down. For some drop-downs, editing or specifying custom values will be necessary. For others, only the specified values are allowed. If you think you need something different, go ahead and try editing (or read the wiki to find out what content is allowed).

Tabs

The Excel spreadsheet contains the following tabs:

NOTE: The order of the tabs is unimportant. Additional tabs can be added and (except as documented under Structure and Code List) will be are ignored. Extra tabs might be useful as play-spaces, locations for extra documentation, places to keep old versions, etc. Within each tab, additional columns can be defined at the discretion of the editor. They will be ignored by the build process. Additional rows other than those containing the defined contents cannot be added. Rows must be contiguous - don't leave blank rows.

Metadata Tab

(**Profile-only**) This tab captures definitional information about the profile - who made it, what it's called, what it's for, etc. This information is not needed for resources and data types because they are not separately maintained and published. The metadata for all FHIR resources and data types is fixed.

- **id**: Required - string. This is a unique id for the profile within the build environment. Normally human-readable, lower-case, dash-separated. If doing a "general" profile, this is the same as the [id] specified in the fhir.ini file. If doing a resource or data type-specific profile, the first word should be the focal resource or data type name followed by a dash followed by one or more qualifying words to create a short, unique, semi-descriptive id.
- **name'**: This is a descriptive name for the profile. If doing a resource-specific profile, this should be the same as the 'Name' for the profile specified in the resource spreadsheet.
- **name.author**: This should be "HL7 International - [owning work group name - e.g. Orders & Observations] WG"
- **name.reference**: This should be the URL for the owning work group on the HL7 website. E.g. http://hl7.org/Special/committees/orders
- **code**: This isn't used at present and should be left blank. If you think you need it, talk to someone from the FMG.
- **description**: This is a short description that will be displayed when displaying lists of profiles. It's also what will show up as the description of the profile if no "introduction" filename is provided below.
- **status**: This should be "draft". It will be changed when the profile goes normative.
- **date**: This should be set to the date the profile was first created. It will be changed when the profile goes normative.
- **published.structure**: This is the name of a tab that defines one of the "published" structures for the profile. It must match the name of one of the tabs in the spreadsheet corresponding to a structure. It must be omitted if the profile only defines extensions and/or search criteria. If a profile defines multiple structures, create multiple rows all with a label of "published.structure" and specify a distinct structure tab name for each. It is possible to have structures that aren't "published". In this case, the tab is defined and referenced from other structures tabs but is not referenced from the metadata tab.
- **version**: This should be omitted unless there are specific reasons to declare a business version for the profile
- **extension.uri**: This provides the base URI for extensions and search criteria as well as for structures referenced in other profiles. It also forms the "identifier" for the profile.
- **introduction**: This is the full name of the "introduction" HTML file (if one exists). This name must be specified for the introduction to be rendered.
- **notes**: This is the full name of the "notes" HTML file (if one exists). This name must be specified for the notes to be rendered.

Data Elements Tab

Resource/Data Type-only - required

This is the tab that defines the content of the data type or resource. Most of the columns used here can be found in the #Element Definition Columns which are shared across multiple tabs. There are, however, three additional columns:

- **Summary**: Optional. This indicates whether the element should be included in a query response for which "summary" elements have been requested. It must be set to either "Y" or "N". (The tool will default it to "N" if omitted.) It is only relevant when defining resources.
- **Regex**: Optional. This may be specified for data elements with 'simple' data types. It asserts a regular expression that should be asserted as part of the schema for validating that particular resource or data type element. (If you want to assert a regular expression on an element in a profile, use an Invariant.)
- **UML**: Optional. This overrides the default placement of the element in the UML layout. It can only be specified on "complex" elements (those without a declared type). There are two different ways the UML diagram can be controlled:
    - The simple way: assert the direction (left|right|up|down) the 'class' should be with respect to the containing 'class'.
    - The explicit way: The format is "x;y", where "x" is the horizontal coordinate in pixels and the "y" is the vertical coordinate in pixels. (The general methodology is play around with values and keep re-generating until the diagram looks the way you want it to.)

Structure Tab

Profile-only - optional

When defining resources or data types, you'll use the Data Elements tab. When defining profiles, you'll use a "structure" tab (or possibly several structure tabs). Structures that are referenced from the metadata tab (as a "Published.structure") will appear as "published" structures that are externally referencable. Those that are not referenced from the metadata tab (but only from other structures) will show up as "non-published". Structure tabs that are not reachable by traversing references from one of the published structures will be ignored.

The name of a Structure is the name of the structure tab. The maximum length of an Excel tab name is 31 (go figure). Because an invariant tab will need to be associated with the structure and the invariant tab name is "[structureName]-Inv", that means that structure names need to be 27 characters or less.

Most of the columns used here can be found in the #Element Definition Columns which are shared across multiple tabs. There are, however, a few additional columns:

- **Profile name**: String - optional. This is a unique name within the structure for a particular row. It only needs to be populated for rows that are defining a 'slice' (see below) or where the element is defining a type whose set of constraints needs to be referenced elsewhere (using the Type column with "@"). However, it may also be populated in other circumstances. If code is generated from a profile, property names will be taken from the Profile name. For that reason, names need to avoid white-space or other characters that would cause code-generation issues.
- **Discriminator**: String - conditional. This can only be specified for elements that have a Profile Name. It must be specified on the *first* slice for a given element path (unless Slice Description is present). E.g. If you were slicing contained observations into three different patterns, the discriminator would be declared on the first slice. This defines the slicing rules for all following sibling elements with the same path. The column conveys three pieces of information: discriminator path, whether slice order matters and the slicing rules. All 3 aspects are separated by "|"
    - The discriminator path is the relative path from the current element to the descendant element that is used to disambiguate between slices. This might be in the same resource or may refer to elements within contained elements or even referenced resources. The path is expressed using dot-notation. E.g. "related.target.name". If there are multiple values, separate them with ",". For guidance on more complex slicing names (e.g. slicing by type or profile, slicing within references, etc.), refer to the spec.
    - The second value is either "true" or "false" with a default of "false". If true, it means that in the instance, slices must appear in the order indicated
    - The final value indicates the rules for slicing. The value may be one of the following:
        - open: (default) - indicates that additional repetitions not matching any of the specified slices may be included in the instance or defined in derived profiles
        - openAtEnd: The same as open, but any repetitions that don't meet the specified slices must appear after the slices defined in this profile
        - closed: Indicates that only the specified slices are permitted. Additional repetitions/down-stream slices are not allowed
    - E.g. "supportingInformation.type|true|openAtEnd"
- **Slice Description**: This must be present if slicing and Discriminator isn't specified
- **Must support**: This can be either "Y" or "N". Default is "N". A "Y" indicates that systems that claim to conform to the profile structure must "support" the data element - the precise meaning of "support" (for both clients and servers) should be defined by the narrative of the profile.
- **Value**: This specifies a fixed value for the element. It may be declared for both simple and complex types. For complex types, the value must be declared as inline XML. Note that the expectation is that valid instances will match the specified XML exactly (no extra elements, extensions or properties.)
    - E.g. for complex type codeableConcept element "<element><coding><system value="system URL"/><version value="ver"/><display value="display"/></coding></element>"
- **Pattern**: This specifies a set of elements and values that must be present in the instance. It should only be present for complex data types and is expressed as XML. Additional elements and values not specified in the pattern are still permitted.
- **Max Length**: Indicates the minimum length that must be supported by all implementers using the element (and thus the maximum length that can be safely exchanged).

Note: While columns might be generally marked as "required", most of these rules are ignored when profiling elements as the definitions are expressed in differential mode. This means that only the Element column and those columns that are being overridden within the context of the profile need to be present in the spreadsheet. Note that though validation rules won't be enforced by the spreadsheet, they still need to be respected. For example, declaring a fixed value if the inherited type includes a choice of multiple data types is still not allowed, even though the data type element might be blank in the profile (because you're inheriting from the resource or an ancestor profile.)

Extensions Tab

Extensions are defined on the Extensions tab. Most of the columns used here can be found in the #Element Definition Columns which are shared across multiple tabs. There are, however, a few additional columns:

- **Code**: String - required. This is the string that must be sent after the extension.uri and the '#' in an instance when identifying this particular extension. E.g. For the extension "http://hl7.org/fhir/Profile/iso-21090#nullFlavor", the "code" would be "nullFlavor". These should generally be lower-camel-case strings. If the extension is part of another extension, the convention is to use string concatenation. E.g. "relationParent.type" as the code for the "type" characteristic of the "relationParent" extension.
- **Context Type**: string - condiditional. This will be one of 3 values: "Resource", "DataType" or "Extension" indicating where the extension is allowed to be used.
    - "Resource" means the extension can be associated with any element in any resource (or any set of elements from one or more resources), including both the root element and complex structures within the resource. (It can also apply to leaf-level nodes within a resource or even to components of a data type, so long as the full context is rooted in a resource.)
    - "DataType" means the extension can be associated with any element that is part of a data type (including the root node of simple data types).
    - "Extension" means the extension can be associated with the root node or some descendant node of a particular extension. Context only needs to be defined for the root node of an extension. Nested extensions (those whose code contains ".") automatically have a context of the parent extension.
    - Note that it is not possible to define an extension whose context can be some combination of resource, data type and extension - multiple extensions must be defined.
- **Context**: String - conditional. This is the list of resource elements, data type elements or extensions this particular extension is allowed to appear in. For resources and data types, this is the full path name of the element. E.g. "Observation.value[x]". For extensions, this is the full path to the corresponding extension this extension is allowed to appear within. Context must be specified (and may only be specified) if Context Type is present.

The following is an example of the definition of a complex extension:.

code	Context Type	Context	Card.
complexExtension	Resource	SomeResource.element	0..1
complexExtension.node1			1..1
complexExtension.node2			0..*

Element Definition Columns

- **Element**: Required - string. This defines the full-path name for the element row. Nodes within the path are separated by ".". E.g. *Patient.contact.name*. Rows must be listed in the order they will appear in the instance. The first row will be the name of the resource or data type. Subsequent rows will be prefixed with the resource name. I.e. All elements listed in a given spreadsheet tab will start with the same prefix.
    - When creating a profile, all path names *must* correspond to names present in the resource or data type being profiled. Note that some paths exist implicitly but aren't listed in the resource spreadsheet: "ResourceName".text, "somepath".extension and "somepath".modifierExtension elements all exist and can be explicitly referred to when creating a profile.
- **Aliases**: Optional. This is a list of comma-separated alternate names for the element. These might be realm-specific or domain-specific or just other common-practice names used in industry. The names here need not have identical scope. The objective is to ensure that someone looking for the resource, resource element, extension, etc. will be able to find the appropriate name by matching on an alias. These names appear in the Formal Definitions tab. For the root resource node, they also appear in the Clinical/Administrative/Infrastructure/Financial tabs
- **Card.**: Required. This indicates the minimum and maximum number of repetitions allowed for the element. For resources, this is constrained to 0..1, 0..*, 1..1 or 1..*. For profiles, any cardinality may be specified, so long as it is a proper constraint on the underlying resource. To prohibit an element, use 0..0.
- **Inv.**: Optional. This is a reference to an invariant that governs the appearance of this element. (I.e. whether the element is allowed to be present or not). It must be one of the values in the "id" column from the #Invariants_Tab. If multiple invariants apply to a single element, they can be separated with ",".
- **Type**: Conditional. This indicates the data type(s) for this particular element. It must be present on the first row of a resource or data element definition (or profile of one) where it indicates whether the structure is dealing with a resource or a data type. It must be present for leaf-level elements (those with no child elements). For complex elements, this is generally left empty, however it may be populated in limited situations
    - First row of resource/data type: For resources, this will be "Resource". For data types, this will be "Structure".
    - Leaf level elements: The allowed value consists of one or more "type" statements. Each type statement includes the defined type (a data type or resource reference). For profiles, it may also include a profile to apply to the data type or resource. For profiles when the type is a resource reference, it may also include an indication of how that resource is conveyed (aggregated)
        - The type is expressed either as the name of a data type (e.g. "string" or "Address"), a reference to one or more resources "Reference(Resource1 | Resource2 | etc.)" or set to "*" to signify that any data type or resource is permitted.
        - A profile can be referenced by following the type with "{" some profile reference "}". E.g. "Address{Address-UK}" or "Extension{#SomeLocalExtension}" or "Extension{http://hl7.org/fhir/Profile/SomeProfile#someExtension}" or "Reference(Patient){Patient-NL}". Note that profiles can *only* be referenced for data types when defining a profile. They can't be referenced when defining a resource or data type. Also, profiles can only be referenced when referencing a single resource. The following would be illegal: "Reference(Patient|Practitioner){Patient-NL)". Instead you'd need to do "Reference(Patient){Patient-NL} | Reference(Practitioner)".
        - When working with complex extensions, sometimes it'll be necessary to refer to an element inside a complex extension (in order to constrain or perhaps extend it). To reference a descendant of a complex extension, specify the URL for the extension followed by a hash sign and the relative path. For example to refer to an extension component author.name.primary, you'd say either "{http..../author#name.primary}" or, if the extension were defined in the same spreadsheet, "{#author#name.primary}".
        - Aggregations can only be specified if the type is a Reference to some resource and if the element is in a profile definition. Allowed values are "contained", "reference" and "bundle". (If you specify "reference", that implies "bundle" as well.) Multiple aggregation types can be declared. If none are declared, then all are presumed to be permitted. So in practice, only the following options make sense:
            - "<contained>" - reference must be to a contained resource
            - "<reference>" - reference must be to a non-contained resource (in bundle or remote)
            - "<bundle>" - reference must be to a resource found in the bundle
            - "<contained,bundle>" - reference must be to a resource either contained or present in the bundle
    - For complex structures, the "type" column is used for one purpose
        - You can say "=Name" to declare what the class name should be for the complex type. This is useful if you want the class name to be different than the association name (e.g. if the class might be referenced from multiple places)
    - For complex structures (which includes profiled simple structures with nested elements):
        - You can say "@some.complex.element.path" or "@[Element Profile Name]" to say that the type for the element should be a complex structure already defined elsewhere within the resource.
- **Is Modifier**: Optional. This indicates whether the element is considered to change the meaning of other elements in the resource/data type. It must be either "Y" or "N". If unspecified, it will be defaulted to "N" by the tool. (Note: This column only exists when defining data types, resources and extensions. It cannot be set or modified in profiles.)
- **Binding**: Conditional. This must be specified for any data element that allows a type of code, Coding or CodeableConcept (including "*"). It should not be present for other elements. It must refer to a Binding name from either the #Bindings Tab in the current spreadsheet or one of the other resource spreadsheets. (Note - the order in which resources are first loaded into memory is controlled by the *build/source/fhir.ini* file. Resources that define bindings must be listed before other resources that use them.
- **Example**: Optional. An example value for the data element. Can only be specified if the type is a single type. Examples can be strings for simple types or XML structures for complex types.
- **Short Label**: Recommended. This is the definition that appears in the XML form as a comment guiding the implementer. If the data type is *code*, the short name *must* be in the form "code1 | code2 | code 3 . . . .". If there are more than 5-6 codes, list the first 5 or 6 and then add a "+". E.g "code5 | code6 +". The short name can be omitted if there's absolutely *nothing* useful that can be said to expand on the meaning of the element name, but usually something should be provided, even if just an example or two. The short name should generally be 50 characters or less.

- **Definition**:Required. This is the full definition of the element. It should not be redundant with either the element name or short name. Examples should be given, if appropriate. This element may be expressed using mark down to allow hyperlinking to other elements in the specification, including bullets and other formatting.
- **Requirements**: Optional. This provides explanation about why the resource, data type or data element within the resource/data type is necessary and/or why it has been constrained as it has. It is optional - only fill it in if the requirements won't be obvious to those with minimal industry experience. (E.g. No need to explain why there's a "name" element on Patient, but Patient.multipleBirth[x] should probably have some explanation.) This element may be expressed using mark down to allow hyperlinking to other elements in the specification, including bullets and other formatting.
- **Comments**: Optional. This element provides additional usage notes associated with the element that are not definitional in nature and don't belong in requirements. These should be element-specific. Usage notes that apply across elements or that span more than a couple of paragraphs should be handled in the usage notes HTML file for the overall resource. This element may be expressed using mark down to allow hyperlinking to other elements in the specification, including bullets and other formatting.
- **To Do**: Optional. This column is for work group convenience. It does not propagate to the resource XML definition though does appear in the specification. We may cause warnings to be spit out for resources that claim to be at DSTU-level with content in this column. This element may be expressed using mark down to allow hyperlinking to other elements in the specification, including bullets and other formatting.
- **RIM Mapping**: Conditional. This contains the mapping of the resource, data type or element to the V3 RIM. The resource level must always be specified, though it can be set to "N/A". If not set to "N/A", all subsequent elements must also be specified, though some of those may also be expressed as "N/A". If present, mappings are generally expressed as pseudo-xpaths. E.g. "Observation[classCode=OBS, moodCode=EVN]". Mappings are expressed relative to the parent element. They are intended for human understandability purposes, not for automated processing or mapping.
- **V2 Mapping**: Conditional. This contains the mapping of the resource, data type or element to the V2 specification (most current release). The resource level must always be specified, though it can be set to "N/A". If not set to "N/A", all subsequent elements must also be specified, though some of those may also be expressed as "N/A". If present, mappings are generally expressed in dot notation. E.g. PID.3.1 They are intended for human understandability purposes, not for automated processing or mapping.
- **??? Mapping**: Optional. Additional mapping columns may also be added. The set of allowed mappings and guidance on required column headings and mapping format is defined in the *build/source/mappingSpace.xml* file.
- **Display Hint**: Optional - This is a set of parameters passed to the narrative generator for this particular element. Each parameter is expressed as "[name]:[value]". Parameter settings are separated by semi-colon. E.g. "param1:value1;param2:value2". At present, there is only one supported parameter:
    - *default*: identifies what the default value is considered to be for this element. If the value in an instance matches the default, it will be excluded from the narrative rendering.
- **Committee Notes**: Optional. This column is for work group convenience. It does not propagate to the resource XML definition nor into the specification. It might include design notes not intended for publication

Invariants Tab

This tab defines constraints that apply to a resource (or to one of the structures defined within a profile. All constraints defined must be evaluatable within the context of the instance, not taking into account any external state information (date, previously received data, other resources, etc.)

- **Id**: This is a unique identifier for the constraint within the Resource/Data Type/Profile. It is used to reference the constraint from data elements whose appearance is controlled by the constraint (if any).
- **Name**: This is a short descriptive name for the constraint. It is used in OCL, Schematron and certain other technologies to display information about the constraint
- **Severity**: 'error', 'warning' or 'best-practice' (the latter being a specialized type of warning)
- **Context**: This indicates the scope (or scopes) in which the constraint should be applied. For data types or resources, this should be the full element path name. E.g. *Patient.contact.name*. For extensions, this should be the full URI of the extension (i.e. the base URI + "#" plus the extension code.) The context determines what the root node is when evaluating the formal expression of the constraint.
- **English**: This is the human-friendly expression of the constraint. It should ideally be appropriate to expose to the end-user (human) of an interacting system.
- **Explanation**: This is required for "best-practice" invariants. It indicates the rationale for the rule
- **Expression**: This is the FHIRPath representation of the constraint. (Required)
- **XPath**: (Optional)This is the formal expression of the constraint expressed using XPath 2.0. There are two namespace prefixes defined: "f:" for fhir and "h:" for xhtml. If you're not familiar with XPath 2, ask Lloyd to write it for you . . .
- **OCL**: (Optional) This indicates an expression of the constraint using the OCL language
- **RDF**: (Optional) This indicates an expression of the constraint using RDF

Search Tab

In resources, there's a single *Search* tab. In profiles, there can be a search tab for each structure (with the tab name [profilename]-search)

The Search tab contains the search operations for a resource or supplemental search operations based on extensions. The standard search operations $page, $count and $id do not need to be specified, but are added automatically.

The Search Tab has the following columns:

- **Name**: Required - a lower-case string. May not end in "-before" or "-after", these suffixes are automatically added for time-based searches.
- **Type**: Required - a choice. Allowed values are:
    - "number"
    - "date"
    - "string"
    - "token"
    - "reference"
    - "composite"
    - "quantity"
- **Target Types**: Conditional - a comma delimited list of strings. Only allowed if Type="reference". If present, indicates that the search only applies to the listed resource types (e.g. if a path called SomeResource.subject is a Reference to multiple resource types, a corresponding search criteria could limit allowed types.

- **Path**: Conditional - The path for the element that is the focus of this search criteria. If specified, must correspond to a path for an element in the resource or must refer to the extension using either "#extensionCode" for a locally-defined extension or the full extension URL for extensions defined in a separate profile. Path is required if Type = "reference" and Target Types are not provided
- **Description**: Conditional - a string. Required if path is not specified, will be defaulted based on path otherwise. This element may be expressed using mark down to allow hyperlinking to other elements in the specification, including bullets and other formatting.

Operations Tab

The Operations tab defines operations related to the resource as well as their parameters.

The rows of this tab are used for both defining operations as well as parameters. The first row (below the heading row) must define an operation. Subsequent rows define the parameters for that operation. Then a second operation can be defined, etc.

For "operation definition" rows, the following columns are used:

- **Name**: Required - The name of the operation (the name by which it will be invoked). Must be a token, starting with a lower-case letter.
- **Use**: Required - One or more of the following values. (If more than one, separate with vertical bars "|".)
    - *system* - The operation can be invoked at the system level, independent of any particular resource
    - *type* - The operation can be invoked at the level of a resource type, independent of a specific resource instance
    - *instance* - The operation can be invoked on a specific resource instance.
- **Type**: Required - Either "query" if the operation is a named query or "operation" if it is another type of operation. (Refer to the OperationDefinition resource for further explanation.)
- **Title**: Required - The descriptive name for the operation - how it's described in the spec.
- **Documentation**: Required - detailed description of how the operation functions. This element may be expressed using mark down to allow hyperlinking to other elements in the specification, including bullets and other formatting.
- **Footer**: Optional - Additional information about how the operation is to be used (will appear in the publication after the formal definition of the operation). This element may be expressed using mark down to allow hyperlinking to other elements in the specification, including bullets and other formatting.

For "operation parameter definition" rows, the following columns are used:

- **Name**: Required - The name of a particular operation parameter. This is expressed as the name of the operation followed by "." followed by the parameter name. Parameter names must be tokens starting with a lower-case character
- **Use**: Required - Must be "in" for input parameters and "out" for output parameters.
- **Min**: Required - A non-negative integer indicating the minimum number of repetitions for the parameter (as input or output).
- **Max**: Required - Either "*" or a positive integer indicating the maximum number of repetitions for the parameter (as input or output)
- **Type**: Optional - Indicates the FHIR data type or search type associated with the parameter. (Can also be "Resource(xxx|yyy|etc.)")
- **Profile**: The full URL of the profile that applies to the parameter.
- **Documentation**: An explanation of the intended content of the parameter. This element may be expressed using mark down to allow hyperlinking to other elements in the specification, including bullets and other formatting.

Events Tab

The use of the Events tab (for defining messaging events) is currently incomplete. If you need to use it, please discuss with the FHIR tooling team (post to the Committers list).

Packages Tab

The Packages tab lists all conformance packages (combinations of structure, extension and/or search criteria definitions) that are associated with the current resource (or data type). This allows the profiles associated with a given resource for linkage in the publication. "Related" means defining extensions, search criteria or structures for the resource/data type.

The tab can contain the following columns:

- **Name**: Required - This is the name of the profile referenced. It will appear in the list of associated profile. (Comment the row by starting the name with '!')
- **Filename**: Required - This is the filename of the profile to be produced in the "publish" directory
- **Source**: Optional - This is the filename of the source for the profile. (Defaults to the same as Filename). It should be relative to the resource directory.
- **Type**: Required - Indicates if the profile is defined as a spreadsheet or a profile XML file. (choices are spreadsheet|profile)
- **IG Name**: Required - the implementation guide that the package belongs to (must refer to an entry in fhir.ini by the code used in the ini file)

Examples Tab

The Examples Tab lists the details for example files for a resource. If the tab does not hold any references to examples, the resource's source directory will be search for a file ending in "{resourcename}-example.xml" and this file is then used as an example instead.

The tab can contain the following columns:

- **Name**: Required - A descriptive label for the example starting with upper-case.
- **Type**, a choice. Allowed values are:
    - (empty) or "xml": Standard XML instance example (could also be XML atom feed)
    - "csv": Requires custom support - ask Grahame if you want to use this
    - "tool": Requires custom support - ask Grahame if you want to use this
- **Description**: Required - Explanation of the purpose of the example and what it shows. This element may be expressed using mark down to allow hyperlinking to other elements in the specification, including bullets and other formatting.

- **Identity**: Required (for XML only) - The identifier of the resource (must meet the constraints of the 'id' data type)
- **Filename**: Required - the name of the example file
- **Profile**: Optional - The name of the profile the example should be linked to. (If blank, it will be linked to the resource)
- **In Book**: If set to "Y", means the example will be included in the book form. (Not relevant right now given that we no longer generate a book form)

Bindings Tab

The Binding tab contains the definition of the Bindings as mentioned in the definition of the data elements. The Binding column in the Data Elements Tab refers to the Binding Name. Bindings may be used in multiple resources, data types and/or profiles, but should only be defined in one spreadsheet.

This tab contains the following columns:

- **Binding Name**: required - a string beginning with an Uppercase letter. Must be unique across the FHIR specification
- **Definition**: required - a string explaining the purpose of this binding. Guides other developers in whether to re-use this binding. Also displayed in the spec.
- **Binding**, a choice. Allowed values are:
    - "code list" - an internal reference to a different tab, which enumerates a simple list of codes
    - "value set" - the name of a file in the same directory as the spreadsheet that has the value set for the attribute
    - "reference" - a direct reference to an external standard (usually an RFC) (not bound to schema. typical examples: language, mime type)
    - "special" - used for infrastructural things by the project team. (usually bound to schema)
    - "unbound" - This should only be used when example codes from an external terminology do not exist and none of the others apply. (This should be VERY rare.)
- **Example** (**Resource-only**) - to indicate the the value set binding is an example. Default is 'N'. Only used with "value set" bindings
    - valid values "Y" or "N"
- **Conformance** (**Profile-only**) - Indicates whether the specified codes must be used. Choices are:
    - "required" - systems SHALL use the codes specified
    - "extensible" - systems SHALL use the codes specified if applicable
    - "preferred" - systems SHOULD use the codes specified
    - "example" - systems MAY use the codes specified
- **max-valueset** (**Profile-only**) - The maximum allowable value set, for use when the binding strength is 'extensible' or 'preferred'
- **Reference**, a string - either #[tab-name], or [valueset-file-name] or [http url] - use depends on the binding column
- **Description**, a string only used when the Binding is a "reference". It provides the descriptive text used for the URL present in the Reference column
- **OID**: Optional - only for code lists. The OID to use for the code list if the FHIR generation process shouldn't assign one
- **URI**: Optional - only for code lists. The URI to use for the code list if the FHIR generation process shouldn't assign one
- **Website/Email**: Optional - only for code lists. The contact information to use for the code list if the default HL7 information shouldn't be used
- **Copyright**: Optional - only for code lists. The copyright information if the default HL7 "public domain" copyright shouldn't be used
- **v2** - the URI of a value set in v2 that the codes are mapped to (binding = code list only). see http://hl7.org/implement/standards/fhir/terminologies-v2.htm for valid values
- **v3** - the URI of a value set in v3 that the codes are mapped to (binding = code list only). see http://hl7.org/implement/standards/fhir/terminologies-v3.htm for valid values
- **Committee Notes** - Additional notes about the selected set of codes - not published

Notes:

- if the data type is "code", you can choose either code list, value set, special, or reference
    - the code choices will be bound by schema. You are allowed to define your own codes in this case, and they are subject to ballot
- if the data type is "coding" or "codeableConcept" you must choose "code list" or "value set" unless the core project team agrees otherwise (can't think of any cases for this)
    - a code list is just converted in a value set reference - it's a cheap and limited way to do an enumeration. Support for this will be removed in the future.
    - whether you use code list or value set, the value set should consist of codes defined elsewhere. Internal codes should not be defined unless suitable codes don't exist elsewhere, and use cases for this should be discussed with the core project team. In principle, these codes are required to be under either harmonisation or external authority

Code list tab

In case the Binding is of type "code list", the column "Reference" must contain the name of another tab containing the valueset for this binding, prefixed by '#'. This tab contains the following columns:

- **Code**: a string - the code that identifies the concept
- **Id**: Optional - a number - used for internal references to create subsumption heirarchies.
- **System**: Optional - a string - used to refer to codes from external systems (use the URI published in the FHIR ballot. if the URI is not published, consult the core project team).
    - note: provide either an ID or a system, but not both
- **Parent**: a string - expressed as "#" followed by the Id value of the code that is the parent of this code (i.e. whose meaning encompasses the meaning of this code)
- **Display**, a string to display this concept to a user. If the code is external, you *should* provide a display so the publishing tools can publish one for user convenience
- **Definition**, a string - mandatory if the code is an internal code. Recommended if the code is external, so the tools can publish it for use convenience.
- **v2** and **v3** - v2 and v3 mappings, if there are mappings defined in the bindings tab (else ignored)
    - the syntax consists of a set of map statements, separated by "," (no spaces)
    - a map statement consists of a single character equivalence, then the table number of code system name, then a ".", and then the code, optionally with a comment in brackets following
    - the equivalent character is "=" - exact. "~" - equivalent. ">" - narrower (i.e. the v2/v3 code is narrower than the FHIR code). "<" - wider
    - e.g. ~0190.H - this code is equivalent to "H" in table 0190

        - e.g. >EntityNameUse.BAD(not sure about old) - this code maps to "BAD" in EntityNameUse, but the definition of "BAD" is narrower. the comment is an explanation of the issue (probably shouldn't be as cryptic as this comment). When you use the equivalence "<", you must make a comment
- **Comittee Notes**, a string - optional. usage guidance..
- In addition, alternate language displays can be defined - any column with a title starting with "Display:" must have a ISO language code in the title - e.g. "Display:fr", and a french equivalent for the Display in the cell for each code