

Sample Tracking Technical Specification

Existing Systems (data)

Identifiers

Per Sample

Sample is the term used to describe a specimen received from a collaborator. The 'Sample' may undergo a number of transformations, such as DNA/RNA extraction, but so long as it is still the same genome it is considered to be the same sample. Multiple IDs are used to track the sample and it's related information across the multiple systems that the sample interacts with during its passage through sequencing. Each system generates its own IDs only when they are needed.

- The ID refers to the genome, not the vial or solution. Changes to the vial and or solution do not change the Sample's ID
- A single sample has only one ID of each type
- Not all of the ID types are created at the same time and not all Sample have all of the ID types.
- Normally the mappings from one ID type to another are considered immutable, there may be rare occasions where manual edits to the data are made to correct a mistake made in loading the data.

IDs for Samples are:

BAC IDs (GLK) Ref_ID, on sample (Lemur)

Bacterial Artificial Genome. Previously these identified a genome fragment that had been made into a BAC for amplification and sequencing. It is now used to refer to a single virus genome, and hence sample.

Source:	Assigned by FLURP or RURP
	Once MRTPCR and or other verification steps have been competed the spreadsheet provided with the samples is loaded into the GLK Extent tables via FLURP or RURP.
Use:	?
Format:	[0-9]{5}
Stored:	Tracker Database
	'Data File' Spreadsheet, after initial submission
	GLK Extent table as 'ref_id' for extents of type 'Sample'
	JIRA

Blinded Number

Source:	Human - Provided in the 'Data File', by the collaborator but based on our specification
Use:	Marked on the aliquots / dilutions taken from the original samples
Format:	[A-Z]{5}_[A-Z][A-Z0-9]*_[0-9]{5}
	From SOP V040 R0.2 p2
	<ul style="list-style-type: none">◦ A 5 letter project id◦ The collection Code◦ The Sample Number
	All separated by underscores.
Stored:	Extent Attribute table, with an Extent Attribute Type of 'blinded_number' for Extents of type 'Sample'.

Extent_id [a subset of]

While each sample has 1 and only 1 extent_id associated not all extent_ids have associated samples, such as 'lot' extents.

- Used to form the tree like view of the data used in Lemur
- Source: GLK
- Use: To encode the structure of the GLK Extent table and to associated Extent Attributes with each other.
- Format: Numeric (20,0)
- Stored: Extent table as the primary key

Key (JIRA)

When samples are added to JIRA as 'Issues' they are assigned an ID by JIRA. Ideally this would be hidden from the user to avoid having yet another redundant id. This is hampered though by several limitations

- The format is fixed
- The issue number cannot be directly set
- The field cannot be hidden
- This is the only field that can be used with 'Quick Search' to jump directly to an issue.

The first three issues would require low level code changes to fix. The final issue, jumping directly to an issue, can be worked around with a modification to search results JSP.

- Source: JIRA
- Use: Required by JIRA
- Format: <JIRA project ID [A-Z]+>-<JIRA issue number[0-9+>
- Stored: JIRA as the primary key for an issue
- Notes: The issue number is not unique across projects

Sample ID (JIRA)

While Key (JIRA) is the actual primary key of the issue in JIRA Sample ID is intended to be the key used for accessing the issues.

- Source: JIRA Issue creator, or manually. It is created from Database, Collection Code and BACid.
Flurp and Rurp have access to all the necessary information and so will be used to create the issue and assign the fields value
- Use: In JIRA to identify a sample
- Format: <Database>_<Collection Code>_<BAC ID>
- Stored: JIRA, custom field 'Sample ID' and the Summary Field.
- Note: Database, Collection Code and BAC ID are to be considered immutable. Collection Code could change in some rare cases but Sample ID should remain unchanged if this does happen.

Tube Label [Weak key]

The identifier written on the sample when it arrives. If the naming convention is followed then this will be a unique id, but when truncated or alternate naming conventions are used it may repeat between collaborators.

- Source: The collaborator with guidance from JCVI
- Use: For identifying the samples when they arrive.

Sample Tracking: Technical Specification

Stored: 'Data File' Spreadsheet
Format: <Collection code> - <Sample Number>

Per Collaboration

Collection Code

Source: Human - Naming scheme unknown. Lemur is used to check collection codes that are already in use within a project.

Use: Used as part of

- Blinded Number
- Sample Name
- Lot Code

Directly used to name the folder that the Initial 'Data File'/spreadsheet will be stored in.
Passed to RURP or FLURP.

Stored: JIRA,
See Blinded Number, Sample Name and Lot Code
Extent as ref_id where the extent type is Collection

Format: [A-Z][A-Z0-9]*

Examples: SAIF, JG, MSLCDC (from SOP V040 R0.2 p2)

Note: The id may end in numbers making it harder to parse a lot code.

Does a collaborator have the same collection code across projects?

Do any collaborators have multiple collaborations. and collection codes?

Per Project

A project is typically a Virus/Eukaryote type or sub-type.

Database Name

Each team/project has its own database instance, which is based on the GLK schema but often contains customizations. Originally each genome had a separate database, but with the addition of simpler genomes, such as for influenza, now multiple are merged into a single database.

Source: Human - The naming scheme is unknown

Use: Required by FLURP, RURP, mh_make_template, makeGelSheet, tracker

Format: The names of these instances are typically three letters

Stored: JIRA
Memorized

Examples: GIV

Notes: The id is assigned before any samples are accepted.

Project ID

Source: Human - The naming scheme is unknown.

Use: Appears in the Blinded_Number

Format: [A-Z]{5}, from SOP V040 R0.2 p2

Examples: TCVSP, RTVSP, MSLSP, MMPSP, RBLSP, VZVSP, ARBSP, ADVSP (from SOP V040 R0.2 p2)

Stored: See Blinded Number

Sample Tracking: Technical Specification

Memorized

Notes: NIGSP for giv, giv3 and swiv.

Possibly GenBank related

CTM (Closure Task Manager)

CTM is the existing sample tracking system. It stores information about the status of samples that are undergoing closure.

CTM Attribute Type

Schema

ctm_attribute_type_id	SMALLINT
name	VARCHAR(50)
type	VARCHAR(10)

Notes

This table, and its foreign key constraint, are used to limit the values used for the 'attribute type' field in ctm_reference_attribute and ctm_task_attribute. It effectively forms an enumerated data type whose values can be modified without changing the schema. This pattern is used repeatedly in the CTM and GLK databases. The other instances of its use will be referred to as Pseudo Enumeration Data types.

Content

The attribute id to name/type relationship changes between the various databases and so the values of ctm_attribute_type_id are not shown. Not all of the databases use all of the attributes, for each attribute the databases that contain it are listed at the end.¹

Name	Type	Database		
Days to Hold	reference	giv	giv3	swiv
Priority	reference	giv		
PCR_Amplicon	task	giv		swiv
PCR_name	task	giv		swiv
PCR_size	task		giv3	
Primer_F_Name	task	giv	giv3	swiv
Primer_F_Sequence	task	giv	giv3	swiv
Primer_R_Name	task	giv	giv3	swiv
Primer_R_Sequence	task	giv	giv3	swiv
Seq_name_F	task	giv		swiv
Seq_name_R	task	giv		swiv

CTM Reference

Schema

Field	Type	Notes
ctm_reference_id	INTEGER	Primary Key
ctm_reference_status_id	TINYINT	Foreign Key to

¹ This table was generated using 'SELECT ctm_attribute_type_id, LEFT(name,17), LEFT(type,15) FROM ctm_attribute_type;' The formatting results in a single entry per line making it easier to turn into a table.

Sample Tracking: Technical Specification

		ctm_reference_status table
type	VARCHAR(10)	
value	VARCHAR(30)	

Content

- type always = 'Extent'
- value = Extent_id
- does status change?

CTM Reference Attribute

Schema

Field	Type	Notes
ctm_reference_id	INTEGER	Primary Key
		Foreign Key to ctm_reference.ctm_reference_id
ctm_attribute_type_id	SMALLINT	Primary Key
		Foreign Key ctm_attribute_type.ctm_attribute_type_id
value	VARCHAR(100)	

Notes

The table requires a JOIN to ctm_attribute_type on ctm_attribute_type_id to form the standard Entity Attribute Value (EAV) table

Content²

Database	Attribute's Name	Values	Occurrences
giv	Priority	0	1058
		45	81
		high	166
	Days to Hold	0	1841
		45	1153
giv3	Days to Hold	0	305
		45	1164
swiv	Days to Hold	0	2065
		45	1

CTM Reference Status

Schema

Field	Type	Notes
ctm_reference_status_id	TINYINT	Primary Key
name	VARCHAR(40)	
display	BIT	

² Find the unique combinations of attribute type & value and then count occurrences

```
select left(name,20), left(value,10), count(ctm_reference_id) from (select ctm_reference_id, name, value from ctm_reference_attribute join ctm_attribute_type on ctm_reference_attribute.ctm_attribute_type_id = ctm_attribute_type.ctm_attribute_type_id) A group by name, value;
```

Sample Tracking: Technical Specification

Notes

Pseudo Enumeration Data type; used in ctm_reference

Content

The actual values vary from database to database, below are the values from giv as an example.

name	Display	name	Display
Received	1	JZ-Avian	0
RT-PCR	0	JS-Avian	0
Sequencing	1	TG-Avian	0
Assembly	1	LO-Avian	0
Closure	0	RA-Avian	0
Manual Edit	0	Resequencing	1
Submission	1	Avian- Unresolved	0
Submitted to Genbank	1	Edit KD	1
Hold	1	Closure KD	1
Published	1	Edit MS	0
Edit-EG	0	Closure MS	0
Edit-TF	0	Edit JH	0
Edit-NM	0	Closure JH	0
Edit-VS	0	Primer Design	0
Edit-JZ	0	Human-Unresolved	0
Rejected	0	Edit TT	1
Reassembly	0	Closure TT	1
Edit-JS	0	Edit MYK	1
Edit-DS	0	Closure MYK	1
Recalled	0	Edit EH	0
Initial Manual Edit	1	Closure EH	0
Validation	1	SJ Closure Hold	0
R&D	0	Coinfection/Mixed	1
Edit-DC	0	Edit BS	0
Pending	1	Closure BS	0
Full Plate Redo	0	Edit SJ	0
Avian Varied Subtype	0	Closure SJ	0
Pending-Extra Contigs	1	FB Post-Validation	0
Edit-AB	0	Deprecated	1
Edit-LO	1	Temp Hold	0
Avian-Mistyped	0	Edit NF	1
Edit-RH	0	Closure NF	1
Edit-TG	0	DW09 Pending	0
Avian-Initial Manual Edit	0	Edit- Becky	1
Final Check	0	454 Sequencing	1
Assembled	0	Draft Submission	1

Sample Tracking: Technical Specification

Assigned	0	NextGen Sequencing Completed	1
Reassembly	0	SISPA	1
Edit-OM	0	Edit JQ	0
Edit-RA	0	Closure JQ	0
Closure-JZ	0	Edit JO	1
Closure-VS	0	Closure JO	1
Closure-JS	0	Edit Javier	0
Closure-TG	0	Closure Javier	0
Closure-LO	1	Unresolved	1
Closure-RA	0	NextGen Validation	1

CTM Task

Schema

Field	Type	Notes
<i>ctm_task_id</i>	<i>INTEGER</i>	<i>Primary Key</i>
assignto	INTEGER	Foreign key from the ctm_user table
creator	INTEGER	
ctm_task_type_id	INTEGER	Foreign key from ctm_task_type Indicates what task needs doing / has been done
ctm_reference_id	INTEGER	Foreign key from ctm_reference A reference to the 'thing' the task is for
ctm_task_status_id	TINYINT	Foreign Key from ctm_task_status
ctm_reference_status_id	TINYINT	Foreign Key into ctm_reference_status table
priority	VARCHAR(10)	Possibly a loosely managed foreign key
creation_date	SMALLDATETIME	
mod_date	SMALLDATETIME	
descr	VARCHAR(255)	A human readable value

Notes

VARCHAR(255) is the largest a field can be in SyBase without using BLOBs and CLOBs which haven't always been supported by the drivers.

CTM Task Attribute

Schema

Field	Type	Notes
<i>ctm_task_id</i>	<i>INTEGER</i>	<i>Primary Key</i>
		Foreign key to ctm_reference table. ctm_reference_id
<i>ctm_attribute_type_id</i>	<i>SMALLINT</i>	<i>Primary Key</i>
		Foreign key to ctm_attribute_type .ctm_attribute_type_id

Sample Tracking: Technical Specification

value	VARCHAR(100)	
-------	--------------	--

Notes

Structurally identical to ctm_reference_attribute other than the first foreign key being from ctm_task instead of ctm_reference. As with ctm_reference_attribute the table requires a JOIN to ctm_attribute_type on ctm_attribute_type_id to form the standard Entity Attribute Value (EAV) table.

Content

	Database		
name ³	giv	giv3	swiv
Primer_F_Name	Format: <segment>_<position><F> Example: NA_1190F		
Primer_R_Name	Format: <segment>_<position><R> Example: NA_980R		
Primer_F_Sequence	Format: [ACGT]+ Example: AGTTTTGTTCAGCATCC		
Primer_R_Sequence	Format: [ACGT]+ Example: ATTTGCTCCATTAGACG		
PCR_Amplicon	Values: No, no, NO, Y, y, Yes, yes, YES	N/A ⁴	None ⁵
Seq_name_F	Format: see <i>Seq name</i> below Example: IXXBA06T00PA05F	N/A	None
Seq_name_R	Format: see <i>Seq name</i> below Example: IXXBA06T00PA05R	N/A	None
PCR_name	Format: see <i>PCR name</i> below Example: IXXX_N9_430F_1240R	N/A	None
PCR_size	N/A	100 to 450	N/A

Seq name:

<lib id ([A-Z]{3,4})>
<unknown ([AB]0[0-9])>
<plate (T[0-9]+)>
<segment ([A-Z]{2})>
<unknown ([0-9]?[A-Z]?)>
<position ([0-9])>
<direction (F|R)>
<version ([B-D]?)>

PCR name:

<lib id ([A-Z]+)> _
<subtype? ([HN][0-9])> _
<forward position ([0-9]+F)> _
<reverse position ([0-9]+R)>

GIV is the only database that contains this attribute, and it only contains four values:

IXXX_N9_430F_1240R
IXYA_N9_10F_550R
IXYA_N9_450F_1000R
IXYA_N9_500F_1015R

3 The name from ctm_attribute_type is used as the ctm_attribute_type_ids have different meanings in different databases
4 This attribute is not present in the above databases ctm_attribute_type table
5 None; the attribute is in the above databases ctm_attribute_type table, but the ctm_task_attribute table contains no references to it.

CTM Task Status

Schema

Field	Type	Notes
ctm_task_status_id	TINYINT	Primary Key
name	VARCHAR(30)	

Notes

Pseudo Enumeration Data type; used in ctm_task and ctm_task_history

Content

The values are the same between the databases:

- Unassigned
- Assigned
- In Progress
- Complete
- Removed
- Unresolved Problem

CTM Task Type

Schema

Field	Type	Notes
ctm_task_type_id	TINYINT	Primary Key
name	VARCHAR(40)	

Notes

Pseudo Enumeration Data type; used in ctm_task

Content

The values vary from database to database.2713

Type	Databases (That contain the given type)		
	GIV	GIV3	SWIV
5' RACE PCR	✓		
Assemble	✓	✓	
Close Gap		✓	
Cover 1x Area	✓	✓	✓
Coverage Analysis	✓		
Edit BAC	✓	✓	
Enter Sample	✓		
Final Check	✓		
Insert Ligated End Sequences	✓		
PCR	✓	✓	✓
Reassemble	✓	✓	
redo RT-PCR sequencing	✓		
Repeat Amplicon	✓	✓	

Sample Tracking: Technical Specification

Resequence	✓	✓	✓
Resolve Ambiguity	✓	✓	✓
Resolve discrepancy	✓		
Resolve frameshift	✓	✓	
RT-PCR	✓	✓	
RT-PCR Amplicon		✓	
SAP	✓		
Sequence	✓		
Submit	✓		
Submit Assembly	✓		
Submit Preview	✓		
Submit Traces	✓		
TOPO Clone	✓	✓	

GLK

Contains extended information about the contigs created from the core data by *Vapour*.

Entity View Attribute

Virtual Table view

These for tables are used to form a virtual logical data store on top of the actual low level database schema. This allows changes to be made to the virtual 'tables' without modification to the schema.

The entries of ExtentType represent the virtual tables.

The entries of Extent represents a row within the virtual tables and has a foreign key to ExtentType to determine which table the row is part of.

The entries of ExtentTypeAttribute represents the columns that may or may not be used. No ExtentType is kept and so any of the columns can be used in any of the tables, although all columns are optional.

The entries of ExtentAttribute represent the actual data of the table. They are linked via foreign keys to ExtentTypeAttribute (column) and Extent (row). They are only able to store data as a VARCHAR(8000).

Object Store view

The structure could also be interpreted as a form of object store. The structure (parent_id) produces the limitation that an object can only be linked to one other object, although one object can have multiple objects linked to it. The graph of objects must therefor be acyclic.

Extent

- Virtual: The entries represent rows within a table.
- Object: The entries represent the objects being stored.

Schema

Field	Type	Notes
Extent_id	Numeric(20,0)	Primary Key
ref_id	VARCHAR(36)	Foreign key to external systems (Not DBMS inforced)

Sample Tracking: Technical Specification

parent_id	Numeric(20,0)	Foreign key on Extent.Extent_id.
Extent_Type_id	INTEGER	Foreign key on Extent_Type.Extent_Type_id
description	VARCHAR(200)	Human Readable

Notes

Extent_id(PK)

Primary key / UID

Each Amplicon, Collection, Genome, Insert, Lot, Sample, Segment and Transposon stored in the GLK has a single corresponding Extent_id

ref_id

Foreign key for external tables and systems. The DBMS doesn't manage the foreign key as it relates to different things depending on which virtual table the row is for. For Samples this is the BAC ID of the sample.

Parent_id(FK Extent.Extent_id)

Creates the hierarchy of the objects stored in Extents.

Virtual: This structure sits above the table metaphore

Object: Hierarchies map easily into the object domain as references between objects

With viruses the Extent_Types should be striated, with all Extents at the same distance from the root having the same type⁶

Other databases do not necessarily follow this convention

The Extent_Types should probably be strictly striated, with each depth being a specific type e.g. Genome, Collection, Lot, Sample, ...

Extent_Type_id(FK Extent_Type.Extent_Type_id)

Virtual: Extent_Type_id links the row to a particular table

Object: Extent_Type_id links the Object to its Class

Content⁷

Extent_Type	giv	giv3	swiv
AMPLICON	383491	167873	194673
COLLECTION	58	26	23

```
6Mapping Extent Types in the Extent heirachy giv3
select distinct(Extent_Type_id) from Extent where parent_id is NULL;
1003      GENOME

select distinct(Extent_Type_id) from Extent where parent_id = 1102843057936;
1002      COLLECTION

select distinct(Extent_Type_id) from Extent where parent_id in (select Extent_id from Extent where
parent_id = 1102843057936);
1006      SAMPLE
1002      COLLECTION
1005      LOT

select distinct(Extent_Type_id) from Extent where parent_id in (select Extent_id from Extent where
parent_id in (select Extent_id from Extent where parent_id = 1102843057936));
1007      SEGMENT
1006      SAMPLE

7 Count uses of each Extent Type in the Extent table
select left(type,30) as type, cont from (select Extent_Type_id, count(Extent_id) AS cont from Extent
group by Extent_Type_id) a join Extent_Type on a.Extent_Type_id = Extent_Type.Extent_Type_id;
```

Sample Tracking: Technical Specification

GENOME	1	1	1
INSERT	338	0	1547
LOT	138	43	59
SAMPLE	4447	1811	2213
SEGMENT	32685	10842	16152
TRANSPOSON	34	0	0

Extent_Type

Virtual:	The entries represent the tables
Object:	The entries represent the classes of object stored in the system.

Schema

Field	Type	Notes
Extent_Type_id	INTEGER	Primary Key
type	VARCHAR(40)	Short Human Readable Name
description	VARCHAR(200)	Human Readable

Notes

Appears to be consistent between databases

There is a full list of the values and their definitions at:

http://confluence/display/VISW/Extent_Type+Terms+Dictionary

Also see 'Everything.doc' for more information of attributes that are not used in sample tracking.

Extent_Type_id (PK)

The UID for the Class/Table represented

type

Human readable name, short.

Virtual: The table name

Object: The Class name

Content

Extent_Type_id	Type	Description	
		Parent	Use
1003	GENOME	-	The root Extent
1002	COLLECTION	COLLECTION GENOME	The ref_id of the extent is the collection code.
1005	LOT	COLLECTION	A group of strains received at the same time. This is not the same a batch, which is a group of samples whose data is released together.
1006	SAMPLE	COLLECTION LOT	The 'Sample' issues in JIRA each correspond to a single SAMPLE in the GLK. (Although not all SAMPLEs in the GLK will have JIRA issues)
1007	SEGMENT	SAMPLE	The system may not need to

Sample Tracking: Technical Specification

1001	AMPLICON	SEGMENT	handle these finer grained parts
------	----------	---------	----------------------------------

The other types are: basura, BAC, GENOMIC_LIBRARY, INSERT and TRANSPOSON.

ExtentAttribute

- Virtual:

The entries represent the data contained in the table.
- Object:

The entries represent the values associated with the Object's properties.

Schema

Field	Type	Notes
Extent_id	NUMERIC(20,0)	Primary Key
		Foreign Key on Extent.Extent_id
ExtentAttributeType_id	INTEGER	Primary Key
		Foreign Key on ExtentAttributeType.ExtentAttributeType_id (Not enforced by DBMS)
value	VARCHAR(8000)	

Notes

Extent_id(FK, Extent.Extent_id) (part PK)

- Virtual:

The row that this data is in
- Object:

The object, stored in the Extent table, that this property belongs to.

ExtentAttributeType_id(FK, ExtentAttributeType.ExtentAttributeType_id) (part PK)

- Virtual:

The column that this data is in
- Object:

Which property is stored in this row.

value

The value stored, owing to the de-normalization the value is always a String

Content

As value contains all of the data for the virtual tables layered over Extent it is a heavily populated table.

ExtentAttributeType

- Virtual:

The entries represent the columns used in the tables. Although they do not indicate which, if any, tables the columns belong to.
- Object:

The entries represent the properties that the Objects might have.

Schema

Field	Type	Notes
ExtentAttributeType_id	INTEGER	Primary Key
type	VARCHAR(50)	Short Human readable name

There is a full list of the values and their definitions at:

<http://confluence/display/VISW/ExtentAttributeType+Terms+Dictionary>

Sample Tracking: Technical Specification

Also see 'Everything.doc' for more information of attributes that are not used in sample tracking.

Content

These are only the types for which further information, beyond that in the confluence page, is known.

ID	Type	Values/Comment
1514	bac_name	NIH022106SARS2
1515	blinded_number	NIGSP_[A-Z][A-Z0-9]+_00[0-9]{3} <project id>_<Collection code>_<sample id> Collection code and sample id checked via db cross reference.
1517	center_project	TIGR_<db name>_<bac_id>
1518	collection	The ref_id of the Extent is used to keep the collection id
1535	created_by	jsitz
	date_to_submit	
	days_to_hold	
1616	deprecated	
1539	disable_auto_assembly	
1565	last_schedule	Sched/Dismi
1567	library_id	[A-Z]{4} (!= collection code)
	regions_to_assemble	
1588	request_auto_assembly	
1589	sample_name	<Collection code (checked)><sample_number (checked)>
1590	sample_number	[0-9]{5}
1606	status	completed/ongoing
1624	is_draft	

Legend	
	Possible Key
	Vapor related
	Set by Workflow
	Not Used

Notes

No data was found in the ExtentAttribute table for the following types:

assembly_seq_count (1512), basura (1538), CEIRS (1516), collection (1518). genotype (1556) , has_(454/illumina/sanger) (1558-1560), inhabiting_organ (1562) , isolate (1564) , lat_lon (1566) , lot_contact_* (1569-1572) , lot_start_date (1573) , notes (1578) , project_id (1585), status_on_assembly (1607), strain_name (1608), virus_name (1613), website_name (1614), source (1621), laboratory_host (1623), project_agreement_id (1655)

Other Data Sources

Custom Primer Spreadsheet

Maintained by Nadia, contains the primer's name, sequence, who is using it...

File System

The next gen sequencing files are stored on a shared file system, as is the initial spreadsheet received with the samples.

The spreadsheets are stored in XXX/<Collection Code>

The 454 sequencing data is stored in XXX as ACE.[0-9]+ files

The illumina sequencing data is stored in XXX as XXX files

/usr/local/projects/GIV/redo_plates.dir

/usr/local/projects/GIV/closure.dir

Contain primers in the format:

<libID><plateID><?[A-Z][0-9]{2}?><Segment, HA, MP, PB2A><offset><[F/R]>

JIRA

Resolution

Jira only counts a task a complete when it has been assigned a resolution. The built in resolutions are

ID	Name	Step
1	Fixed	
2	Won't Fix	Unresolved, Failed
3	Duplicate	
4	Incomplete	
5	Cannot Reproduce	
7	Not A Bug	Deprecated
8	Work Complete	Sample Published, Complete

The resolution is normally set by the user, but for sample tracking the workflow state indicates what resolution to use. It can be set using Set field Value on Transition on page33

Resolution can only be cleared via the workflow, no null/blank/unset option is provided when editing. The field is cleared, and will display as 'unset', by setting it to the empty string.

Existing Systems (tools)

autotasker2

Source: Elvira

Checks for various flaws in the data, see spec. Produces a list of any problems found, and for NG a '.nav' file to help locate them in consed. The results of autotasker2 are currently manually interpreted to assign a category (Complete, missing regions, ...) to a sample.

Should be started via Java Commons on the Grid.

CTM

CTM maintains a series of 'queues'/pools that samples can be associated with. It enforces the constraint that a sample can only be in one pool at any given time. To simulate the ability to assign tasks to individuals queues have been created that are a product of the users and the states, e.g. Mr X stage 1, Mr X stage 2, Ms Y stage 1, Ms Y stage 2.

CTM provides the ability to see information on the plates and amplicons and set which plates, referenced in the format #[0-9][0-9], and amplicons to use in closure.

CTM's current uses:

CTM provides a method of selecting plates and primers for closure.

Notes:

CTM lists the 'bins' down the left of the screen with drop downs below any bin that contains samples. The drop downs list all of the samples in that 'bin'. They are listed in the format

<BACid> (<reference>)**

Reference is a code unique to CTM. There is a Quick Launch function that takes the reference number and opens that record. It can also work using BAC IDs if they are prefixed by 'BAC:'.

When a sample is selected the following information is displayed

- Reference The CTM reference
- Status ctm_task -> ctm_task_status
- Extent Extent id
- Homepage for BAC BAC ID

BAC Info

- Host Extent Attribute 'host'
- lib_name Extent Attribute 'samplename'
- lib_desc ?

BAC Libraries

- lib_id Links to a popup of Tracker data
- min ?
- med ?
- max ?
- entry_pn a user name
- Splice_name e.g. PCR-TOP02.1TA
- cont. list Contaminants, e.g. E. coli

ctmbacs

File: /usr/local/devel/CLOSURE/ctm/script/ctmbacs

Usage message

ctmbacs is a BAC/Reference management tool for the CTM, that allows status viewing and updating on the command-line. Updates any combination of BAC status and log comments.

```
usage:
  ctmbacs -D <database> [options]

options:
  -U      username
  -P      password
  -S      server
          [default: SYBTIGR]
  -p      passwordfile
  -b      list of bac_ids separated by comma or "-". e.g. 14046-14052,14891
  -B      file of bac_ids

  -status_codes
          flag to print CTM status codes for a project

  -report
          when supplying a status code (number or text), displays
          bacs in that status code. When no status code is supplied,
          counts bacs by status.

  -update
          flag used to update status and/or log entries
          requires one or more update_options (below)
```

```
update_options:

  **all update options must be accompanied with the -update flag

  -c      used with -update, enter CTM BAC comments on the command line
  -C      used with -update, enter CTM BAC comments in a file
  -status
          used with -update, enter CTM status as a code or text description
```

Example usage:

- 1. Show status of bacs for giv (format: bac_id status_code status)
 - a. all bacs : ctmbacs -D giv
 - b. certain bacs: ctmbacs -b 14046,14047
- 2. update bacs 14046,14047 status with comments in a comment file, mycomments
 - a. Using text status: "Submitted to Genbank"
ctmbacs -D giv -U hkoo -b 14046,14047 -C mycomments -status "Submitted to Genbank" -update
 - b. Using status_codes (not recommended):
ctmbacs -D giv -U hkoo -b 14046,14047 -C mycomments -status 8 -update
- 3. update reference 10,15-23,37 status with comments in a comment file, mycomments
 - a. Using text status: "Submitted to Genbank"
ctmbacs -D giv -U xliu -r 10,15-23,37 -C mycomments -status "Submitted to Genbank" -update
 - b. Using status_codes (not recommended):
ctmbacs -D giv -U xliu -r 10,15-23,37 -C mycomments -status 8 -update
- 4. Show all bacs in status Submission (format: bac_id status_code status)
ctmbacs -D giv -report Submission
- 5. Show counts of all bacs by status (format: Status count)
ctmbacs -D giv -report
- 6. Show all CTM reference status and their ids
ctmbacs -D giv -U xliu --status_codes

Output format

For 'Show all bacs in status X' the output format is:

```
<bac_id> <status_code> <status>
```

For 'Show all bacs by status' the output format is:

```
<status> <count>
```

For 'Show all CTM reference status' the format is:

```
statuses and their ids
```

Flurp / Rurp

Ask Palo which source is the most up to date

Flurp is an influenza specialized version of *Rurp*
used for the initial loading of samples into the project db.

Lemur

Source: Elvira

GLK interface

The sample can be directly accessed using the url:
viewExtent.php?type=SAMPLE&ref=36123

Note: The user needs to already be logged into Lemur for this to work, logging in
redirects to the index page (not the requested page)

Vapor

Source: Elvira

Transfers sequencing data into GLK and initiates the assembler once a programmed quantity of reads have been transferred into GLK, the number is normally the expected number of reads - a safety margin to allow for some failed reads. When the assembler is not wanted, as is the case when loading closure sequences for a NG sample, the number of pieces to wait for is set to a number significantly larger than the number of reads.

- Currently uses FLAP which will be replaced with CLC.
- Requires environment variables to be setup to allow it to use the Grid.

Flu Validator

Source: ?

Uses NCBI's FLAN

Used by Autotasker2

Replaces Flava

3 possible outputs

- Failed - If FLAN returned any errors
- Draft - If the sequence is only partial or contains ambiguities
- Valid - If it is neither Draft nor Failed

Flava

Replaced by Flu Validator

- Can't handle draft
- Uses out of date submission rules

Old Submission Criteria (new undecided)

Less than 5 Ambiguities

Less than 20bp Missing

Max missing at the start, 9bp

Max missing at the end, 12bp

Only the start or the end may have missing bases, not both

Migration

Linking between the systems

The new system will need to communicate with the GLK.

GLK

Extent_id is used in the GLK as the primary key to the entities table (Extent). Each sample known to they system has one and only one entry in this table but not all rows in the table represent samples. The extent_type field is used to indicate what type of entity a row 'contains'.

CTM

ctm_reference_id is used as the primary key to the entities table, ctm_reference, that contains samples in the CTM. The rows in ctm_reference don't provide a means of directly determining the entities type, it would just be replicating the information from the GLK. Instead they provide the extent_id in the value field which can be used to join it with the extent table.

CTM < > GLK

```
select value from ctm_reference where ctm_reference_id = <ctm id>;
select ctm_reference_id from ctm_reference where value = '<extent id>';
```

JIRA

Jira uses it's own ids for issues. They are in the form <project>-<issue number> e.g. ST-137. Atlassian resolutely refuse to allow external identifiers to be used, so it is not possible to create and issue with an id or even just id number that matches the extent id. The extent id will have to be stored in a separate field which while awkward can be worked around. This unfortunately makes mapping from an extent id to a JIRA issue much more awkward. The CLI tool can only use the JIRA ID to access the Issue. As the mapping between extent_id and JIRA_ID is effectively random it is not possible to calculate the mapping, instead it must be looked up.

Quantity of Data

For influenza there are three Viral Genome Databases (VGDs); giv, giv3 and swiv. These each house seperate instances of the CTM. The age and quantity of samples in each of these CTMs varies:

Database	Age ⁸	Size ⁹ Total References (Total Tasks)
giv	December 2004	4,344 (28,193)
giv3	December 2006	2,148 (10,330)
swiv	May 2009	2,303 (4,214)
Total	Oldest is from December 2004	8,795 (42,737)

JIRA currently contains 16,297 issues. If a new issue was created for each CTM Reference and CTM Task the number of issues in the system would almost quadruple.

The number of new issues could be cut by storing CTM Tasks as comments instead of Sub-Tasks. Converting Tasks reduces the number of new issues to about 8,795 (a 50%) increase.

Limiting the age of the CTM References migrated would also help reduce the quantity of new issues.

Not migrating the data avoids any space concerns but requires the CTM to be left up, and kept functional, in case old information is needed.

If the number of historic samples that need examining is small it would make sense to convert the samples only if and when they were needed.

8 Done using 'SELECT MIN(date_in) FROM ctm_reference_history;' This is actually the date that a 'CTM referenced' object was first updated, but it will be very close to the first use of the db.

9 Values as of June 16th 2011. Collected using 'SELECT COUNT(*) FROM ctm_reference;' and 'SELECT COUNT(*) FROM ctm_task;'

JIRA Plug-ins and Extensions

Todo: description of the types of plug-ins available with JIRA.

Custom Field Plug-ins

Migrating an Existing Field

The GUI doesn't provide a way to do this, but it can be done via the database. The data may need to be copied into a different column of the customfieldvalue table, if the underlying data type used for storage changes.

This example from Atlassian's site checks both the current field name and type, presumably to avoid any records that have already been converted.

```
UPDATE customfieldvalue SET textvalue=stringvalue
WHERE customfield=
  (SELECT ID FROM customfield WHERE
    customfieldtypekey=
      'com.atlassian.jira.plugin.system.customfieldtypes:textfield' AND
    cfname='Text Field'
  );
```

The class used for a field, its type, is defined in the customfield table.

```
UPDATE customfield
SET CUSTOMFIELDTYPEKEY=
  'com.atlassian.jira.plugin.system.customfieldtypes:multiuserpicker',
CUSTOMFIELDSEARCHERKEY=
  'com.atlassian.jira.plugin.system.customfieldtypes:userpickersearcher'
where
  cfname='MyMultiSelect';
```

The values for the types are not actually class names. They are the plug-in identifiers, their values' come from the plug-in definition file. Specifically:

<plug-in key>:<customfield key>

<plug-in key>:<searcher key>

where the values for the keys come from:

- <atlassian-plugin key="plug-in key"
- <customfield-type key="customfield key"
- <customfield-searcher key="searcher key"

Custom Field Type Interface

Storage / Type conversion

CustomFieldType contains the hooks for controlling storage. There are several encodings used for the values:

- Persistence Objects: Normally Strings, other basic types like Long are also supported
- Field Objects: These can be of any type and are the 'Natural' representation of the value.
- HTML Parameters: This is the format used to pass the value from the form on in the Edit page back to the server.

There are methods in CustomFieldType to handle converting between them as well as methods to store and retrieve the values for an issue or the default value:

Sample Tracking: Technical Specification

Method	Input Object Type	Output Object Type	Notes
getSingularObjectFromString	Persistence Object	Field Object	
getDbValueFromCollection	Field Object+	Persistence Object+	Optional; not part of the interface but commonly implemented.
getValueFromIssue	-	Field Object	Retrieves Persistence Objects and converts them before returning.
	null	Field Object	Get the default value
createValue / updateValue	Field Object / Field Object+	-	Converts the input to Persistence Objects before storing them
remove	-	-	Called to remove the custom field, including any data stored.

Persistence (CustomFieldValuePersister)

The values are stored and retrieved by *CustomFieldValuePersister*. The methods accept and return Collections of Persistence Objects. The key used to store and retrieve values needs to be based on the Custom Field instance and a unique identifier (Issue.getID). A type for the Persistence Objects is also required, generally PersistenceFieldType.TYPE_UNLIMITED_TEXT is used.

The persister is used by the following Custom Field Type methods:

Custom Field Type Method	Custom Field Value Persister Method
getValueFromIssue	getValues
createValue	createValues
updateValue	updateValues
remove	removeAllValues(customField.id)

When the issue is null a special value should be used to indicate that the default is being stored/retrieved.

Defaults

The default is stored using GenericConfigManager. The config manager stores and retrieves based on two keys (strings). The first is set to CustomFieldType.DEFAULT_VALUE_TYPE. The second is set to FieldConfig.getId().

```
public void setDefaultValue(FieldConfig,Object)
```

Essentially the same as create/update value but with different parameters passed to CustomFieldValuePersister

```
public Object getDefaultValue(FieldConfig,Object)
```

The object returned is a collection of objects for the field's type.

HTML Form creation

All interaction with the field is done by the user is carried out through the web interface, using HTML. The HTML is generated by velocity templates:

Create template

To allow a user to set the initial value for a field an HTML form element must be generated to capture that information. This is done by the create template, this can be the same template as that associated with the edit operation.

Edit template

To allow a user to edit a field also requires a HTML form element to capture the value selected. Unlike the create operation the field may already have a value and this needs to be presented to the user. The template provides the formatting of the value and the 'getVelocityParameters' method is used to pass the Field Object to the template.

View template

Column view template

public void DatabaseValuesCFType.getVelocityParameters(Issue ...

When any of the templates are going to be used this method is called first to allow the field to retrieve a value to place in the environment for the template. The Issue value is null for the creation operation. As with other methods a null Issue can be considered a request for the default value.

HTML Form processing

The CustomFieldParams objects contain the values from the HTML. The keys, and hence values, available will only be those that match the custom field id, 'customField_10000' etc. A Map of the keys and values is available via getKeysAndValuesbut as there is only one key it is more normal to call getAllValues which returns a collection of just the values.

Multiple fields can be used by assigning the same name to multiple inputs. The resulting values can be retrieved, in the order in which they were in the form, by iterating over the collection returned by getAllValues.

public void validateFromParams(CustomFieldParams, ErrorCollection, FieldConfig)

Receives the form's values as name value pairs in the CustomFieldParams object.

*public Object getValueFromCustomFieldParams(CustomFieldParams) throws
FieldValidationException*

Converts from the name value pairs of the form into the object type of the field. If there are problems which make this impossible then a FieldValidationException is thrown.

public String getStringFromCustomFieldParams(CustomFieldParams)

Converts from the name value pairs of the form into the object for the velocity template, accessed as \$value. In the multi-value example a collection is returned, so it would appear that the string in the method name is optional.

Other

public String getChangeLogValue(CustomField, Object)

The object is a collection of objects of the field's type. The String returned is added to the change log. There doesn't seem to be a way provided to find out what the previous value was and so the messages must be of the form X now equals Y.

Further methods could be present if the class also implements one or more of:

- ProjectImportableCustomField
- SortableCustomField

DatabaseValuesCFType.getVelocityParameters(Issue ...

Is a convenient way to pass information/objects to the velocity template

Field Indexer Interface

The indexer translates between the value stored in the persistence framework and the value that should be used when searching. There is an abstract implementation of FieldIndexer, AbstractCustomFieldIndexer that can be used as the basis of custom implementations.

FieldIndexers only requires 4 methods:

Sample Tracking: Technical Specification

```
String getId();

String getDocumentFieldId();

void addIndex(lucene...Document, Issue);

boolean isFieldVisibleAndInScope(Issue);
```

AbstractCustomFieldIndexer's constructor takes a CustomField parameter, bean like getters are provided to access it. that can be used. It supplies one utility method

```
protected boolean isRelevantForIssue(Issue)
```

AbstractCustomFieldIndexer implements all of the methods of FieldIndexer but adds two abstract methods for sub-classes to implement:

```
public void addDocumentFieldsSearchable(lucene...Document, Issue);

Fields added to the doc should be created with index type Field.Index.UN_TOKENIZED

public void addDocumentFieldsNotSearchable(lucene...Document, Issue issue);

Fields added to the doc should be created with index type Field.Index.NO
```

The only difference between the two is the field type used when creating the Field object. The implementations are only passed the Issue and so need to keep a reference to the field to retrieve the value for indexing. ?

The implementations add their values to the index by calling:

```
((lucene...Document) doc ).add(
    new Field(getDocumentFieldId(), <value>, Field.Store.YES, <index type>));
```

Velocity Environment

Object	Description	Edit	Views
action Not Needed for view	<i>unknown</i> Calling action.	✓	X
applicationProperties See Description	<i>com.atlassian.jira.config.properties.ApplicationProperties</i> Via constructor or ComponentManager	✓	✓
authcontext Via constructor	<i>com.atlassian.jira.security.JiraAuthenticationContext</i> for authentication information	✓	✓
baseurl	<i>String</i> The getContextPath of the req object NOT AVAILABLE	✓	✓
buildutils Empty constructor	<i>com.atlassian.jira.util.BuildUtils</i> has information on build numbers, editions etc. (deprecated)	✓	✓
config Not Needed for view	<i>com.atlassian.jira.issue.customfields.config.CustomFieldConfig</i>	✓	X
configs Not Needed for view	<i>unknown</i> The various configuration items for that context.	✓	X
constantsManager See Description	<i>com.atlassian.jira.config.ConstantsManager</i> Object for managing "constants" (issue types, resolutions...) Via constructor or ComponentManager	✓	✓
customField From parameter	<i>com.atlassian.jira.issue.fields.CustomField</i> Information on the current field.	✓	✓
customFieldParam	<i>com.atlassian.jira.issue.customfields.view .CustomFieldParams</i>	✓	✓

Sample Tracking: Technical Specification

s See Description	This is where the value is pulled from, for convenience (CustomFieldParams) customField.getValue(issue)		
dateutils Empty constructor	com.atlassian.core.util.DateUtils utilities for dates	✓	✓
descriptor	CustomFieldTypeModuleDescriptor The module descriptor of the current field CustomFieldType.getDescriptor()	✓	✓
displayParameters Not Needed for view	unknown Custom parameters to the template, such as whether to display headers	✓	X
fieldLayoutItem Not Needed for view	com.atlassian.jira.issue.fields.layout.field.FieldLayoutItem isRequired, isHidden etc	✓	X
i18n See Description	com.atlassian.jira.web.bean.I18nBean utilities for internationalization JiraAuthenticationContext.getI18nHelper()	✓	✓
issue From parameter	com.atlassian.jira.issue.Issue Access to the issues properties	✓	✓
jirakeutils Empty constructor	com.atlassian.jira.util.JiraKeyUtils utilities for parsing keys	✓	✓
jirautils Empty constructor	com.atlassian.jira.util.JiraUtils general utilities	✓	✓
outlookdate See Description	com.atlassian.jira.web.util.OutlookDate for formatting dates, JIRA style Construct with Locale	✓	✓
projectManager See Description	com.atlassian.jira.project.ProjectManager Via constructor or ComponentManager.getInstance().getProjectManager();	✓	✓
req	HttpServletRequest NOT AVAILABLE	✓	✓
textutils Empty constructor	com.opensymphony.util.TextUtils utilities for text manipulation needs	✓	✓
userutils Empty constructor	com.atlassian.core.user.UserUtils utilities for getting users	✓	✓
value From persistence framework	String value of the custom field. Other types are used by other fields	✓	✓
velocityhelper Empty constructor	com.atlassian.jira.util.JiraVelocityHelper general utilities (deprecated)	✓	✓

[move] All but 'baseurl', 'descriptor' and 'req' can be provided to the instance creating the value to store!

Field Configuration Interface

The custom field declares what its current configuration is and how to edit the values via the FieldConfigItem. I think that it is intended that each configurable property has a FieldConfigItemType object because the interface only provides for one name and one value to be displayed. It seems more sensible however to use one per group of properties and to display a summary for the value, otherwise the user has to change page to edit each parameter in turn.

FieldConfigItemType

com.atlassian.jira.issue.fields.config.FieldConfigItemType

String getBaseEditUrl()

- Java doc: Creates the base of the URL that links to the configuration screen for this type of configuration.
- Example: "EditXXXConfig.jspa"

Object getConfigurationObject(Issue, FieldConfig)

- Java doc: Returns an Object that holds the the configuration.
- Example: Map results = new HashMap();
results.put("config1", retrieveStoredValue(<FieldConfig>));
return results;
- Notes: This is passed to the the Velocity template via \$configs

String getDisplayName()

- Java doc: The user interface name for this dimension of configuration.
- Example: "Database"
- Notes: The name seen in the screen to select the type of configuration to set, e.g. default

String getDisplayNameKey()

- Java doc: The i18n key for the user interface name for this dimension of configuration.
- Example: "Selected Database"
- Notes: Appears in the field configuration screen (view) as the field name. e.g.
Selected Database: ...




String getObjectKey()

- Java doc: Returns a key unique among FieldConfigItemType implementations for the configuration value so it can be retrieved from a key-value pair store.
- Example: "DatabaseConfig"
- Notes: Used with \$configs to retrieve the object from getConfigurationObject

String getViewHtml(FieldConfig, FieldLayoutItem)

- Java doc: Renders a view of the current configuration as HTML.
- Example: String value = retrieveStoredValue(fieldConfig);
return getAsHtml(value);
- Notes: Used with DisplayNameKey to display key value pair.

The implementing class is normally called <Plug-in name>ConfigItem

Default Configuration Scheme for VCF		 
Default configuration scheme generated by JIRA		
Applicable contexts for scheme:	Global (all issues)	 Edit Configuration
Default Value:	;	Edit Default Value
DisplayNameKey:	getViewHtml	Edit DisplayNameKey

Persistence (configuration)

The values are stored in a properties set, which is accessed by passing a map of name value pairs to `PropertySetManager.getInstance(table name, map)`. The table is "ofbiz" and the map contains:

Key	Example Value	Notes
delegator.name	"default"	
entityName	"db_fields"	Unique to the 'type' of properties set
entityId	Long(1)	identifies a particular instance of type entityName

The example given doesn't use the entityId, keeping it a constant value and then stores the separate contexts all in the same properties set, but with unique keys. The keys are generated by:

```
String entityName = fieldConfig.getCustomField().getId() + "_" +
                    fieldConfig.getId() + "_config";
```

The actual reading and writing of values is done by calling `getString(entityName)` and `setString(entityName,value)` on the `PropertySet`

Multiple Instances

`com.atlassian.jira.issue.fields.config.FieldConfig`

FieldConfig represents one instance of a custom field. There could be multiple instances in one project and or across multiple projects. The methods of FieldConfigItemType accept a parameter of type FieldConfig to distinguish which instance is being used.

Retrieving the correct configuration in a CustomField implementation:

```
Issue issue, CustomField field
// First get the fieldConfig
FieldConfig fieldConfig = field.getRelevantConfig(issue);
// Then use it in the same way as in the FieldConfigItemType class
String config = retrieveStoredValue(fieldConfig)
```

Viewing (Configuration)

FieldConfigItemType contains three methods that control how the currently set value of the configuration is displayed:

```
String getDisplayName()
    Defines ?

String getDisplayNameKey()
    Defines what, internationalizable, key to display when viewing the setting

String getViewHtml(FieldConfig, FieldLayoutItem)
    Provides the HTML that will be inserted in the table to represent the value configured.
```

Editing (Configuration)

FieldConfigItemType contains three methods that setup the environment for velocity to render the edit page:

```
String getBaseEditUrl()
    Sets which template to use

String getObjectKey()
    Used when placing the configuration object into the velocity environment as part of the map $configs

Object getConfigurationObject(Issue, FieldConfig)
    Used to place the configuration object into $configs
```


IssueTabPanel

The tabs at the bottom of the Issue View can be added to by objects that implement the IssueTabPanel interface.

- `init`: Initializes the plug-in, passing it an IssueTabPanelModuleDescriptor based on the XML used to define it in the atlassian-plugin.xml file
- `getActions`: Returns an un-typed list containing objects that implement the IssueAction interface. These objects do the actual HTML generation. The Issue and the User objects are passed to this method, to pass on to the IssueAction objects.
- `showPanel`: Returns true if the tab should be included in the list of tabs based on the Issue and the User.

The IssueAction interface contains:

- `getHtml`: Renders the HTML for this subsection of the Issue Tab Panel
- `getTimePerformed()`: Returns a date that is used in the 'All Tab' both for displaying the Issue Action and for ordering the Issue Actions.
- `isDisplayActionAllTab()`: Returns true if this Issue Action should also appear on the 'All Tab'.

Note: None of IssueAction's methods take parameters. The IssueTabPanel will need to pass anything they need to them at creation, before they are passed to the system via `getActions`.

actions.jsp includes the panels with:

```
<webwork:iterator value="/issueTabPanels">
  <webwork:if test="/page == completeKey">
    <li class="active"><strong><webwork:property value="label" /></strong></li>
  </webwork:if>
  <webwork:else>
    <li>
      <a href="
        <webwork:url value="'/browse/' + issue/string('key')">
          <webwork:param name="'page'" value="completeKey"/>
        </webwork:url>
      ">
        <strong><webwork:property value="label" /></strong>
      </a>
    </li>
  </webwork:else>
</webwork:iterator>
```

The dynamic parts are in **bold**. After this block the actions are added, but these are only taken from the open tab. The label appears to be the only part of the tab that is always rendered. The label is HTML encoded (e.g. `<` becomes `<`) and so can't be used to insert anything but text.

JSPs

Much of the GUI of JIRA is generated with JSPs. The JSPs used are accessible in the installation directory.

Advantages

- They provide a direct way of altering the behavior of the interface

Disadvantages

- They are overwritten when a new version is installed.
- Plug-ins cannot include JSPs as they will not get compiled

Pluggable Footer

Pluggable Footer Objects insert a block of HTML into the footer of every returned page.

There is a default implementation provided by Atlassian, `DefaultPluggableFooter`. It renders a velocity template (defined as 'view' in the atlassian-plugins.xml file).

Example

The footer plug-in used to produce the current footer in JIRA is a good example. The plug-in definition is in:

```
WEB-INF/classes/system-footer-plugin.xml
```

The plug-in definition is:

```
<atlassian-plugin name='JIRA Footer' key='jira.footer' i18n-name-key="footer.plugin.name">
  <plugin-info>
    <description key="footer.plugin.desc">
      This plugin renders the content of the footer in JIRA.
    </description>
    <vendor name="Atlassian" url="http://www.atlassian.com"/>
    <application-version min="3.12" max="3.12"/>
    <version>1.0</version>
  </plugin-info>

  <jira-footer key="standard-footer"
    i18n-name-key="footer.plugin.standard.name"
    name="Standard Footer"
    class="com.atlassian.jira.plugin.navigation.DefaultPluggableFooter"
    system="true">
    <resource type="velocity"
      name="view"
      location="templates/plugins/footer/footer.vm"/>
    <order>10</order>
  </jira-footer>
</atlassian-plugin>
```

Key

The sections that will need to be changed are highlighted.

The boiler plate parts are grayed out.

Notes

The footer is included even when the user is not logged in. The code/template should check for a valid user, or if any object it needs from the environment exists and has been setup before adding itself.

API

Beyond the default implementation custom a class that implements PluggableFooter can be used. The methods that need to be implemented are:

init: Initializes the plug-in, passing it an IssueOperationModuleDescriptor based on the XML used to define it in the atlassian-plugin.xml file

getFullFooterHtml: Generates the HTML for pages where width='100%'

getSmallFooterHtml: Generates the HTML for other pages

Both get HTML methods are passed an HttpServletRequest object.

Velocity Context

The default implementation takes a velocity template as a parameter. That template gets its own context, separate from the rest of the page. The context contains:

Information on the software and installation

Name	Type	Example
buildInformation	String	4.0#466
serverid	String	AR36-DKLC-29S2-IZVC
version	String	4.0
buildNumber	String	466
edition	String	enterprise

License Information

A range of License properties, only one of which will be present at a time. If they are present their value will always be Boolean.TRUE.

Sample Tracking: Technical Specification

The properties licenseMessageClass and longFooterMessage's values are determined by the type of license present. The license properties, along with their associated values for licenseMessageClass and longFooterMessage, are below.

Name	licenseMessageClass	longFooterMessage
unlicensed	licensemessagered	TRUE
evaluation	licensemessagered	FALSE
confirmedWithOldLicense	licensemessagered	TRUE
community	licensemessage	FALSE
opensource	licensemessage	FALSE
nonprofit	licensemessage	FALSE
demonstration	licensemessage	TRUE
developer	licensemessage	TRUE
personal	licensemessage	TRUE

Utility Class Objects

Name	Type
externalLinkUtil	com.atlassian.jira.web.util.ExternalLinkUtil
utilTimerStack	com.atlassian.util.profiling.UtilTimerStack

Other

Name	Type	Notes
req	HttpServletRequest	The same request passed into the footer plug-in
organisation	String	'Atkassian' on the demo copy
smallFooter	Boolean	Always TRUE if present. It is only added if the getShortFooterHtml method was called.

HttpServletRequest

The plug-in's API defines that the get HTML methods receive a HttpServletRequest object. That same object is passed to the velocity environment as 'req'. The contents of the request and the session are detailed in the reference document 'HttpServletRequest contents'.

One use of this object is to determine what the current project is, to limit the impact of introducing a new plug-in on other projects.

```
JiraHelper(HttpServletRequest).getProject();
```

Pluggable Issue Operation (4.0 and earlier only)

The list of operations in the left-hand column of the view issue page is generated from plug-ins in versions 4.0 and earlier of Jira. The interface is only three methods:

- init: Initializes the plug-in, passing it an IssueOperationModuleDescriptor based on the XML used to define it in the atlassian-plugin.xml file
- getHtml: Produces the HTML to display, based on a passed in Issue.
- showOperation: Controls the inclusion of the item based on a passed in Issue.

Pluggable Top Navigation

Pluggable Top Navigation is used by JIRA to add the strip menu below the logo. The interface contains only two methods:

- `init`: Initializes the plug-in, passing it a `TopNavigationModuleDescriptor` based on the XML used to define it in the `atlassian-plugin.xml` file, or `system-top-navigation-plugin.xml`.
- `getHtml`: Renders the HTML for inclusion before the main page

Web Fragments (Limited use in 4.0)

There are three types of fragment;

- Web Panel (Jira 4.4+): for adding new sections
- Web Item (Jira 3.7+) : for adding to existing bottom level menus
- Web Section (Jira 3.7+) : for adding new drop-down menus

Despite Web Item and Web Section having been added several versions ago they only have access to a few menus in Jira 4.0, the main top bar menu and the projects menu.

Workflow Condition Plug-ins

Existing

SubTaskBlockingCondition

Format

```
<condition type="class">
  <arg name="statuses">[0-9]{5}([,][0-9]{5})*</arg>
  <arg name="class.name">...condition.SubTaskBlockingCondition</arg>
</condition>
```

Description

Statuses is a list of the reference IDs of the statuses that the Sub-Tasks must be in before the action protected by this condition can take place.

Custom Condition Plug-ins

Use the same format for each plug in types description

Factory:	WorkflowPluginConditionFactory
Methods:	getDescriptorParams(Map<String,Object>) : Map<String,Object> getVelocityParams(String resourceName, AbstractDescriptor): Map<String,Object>
Implementation:	com.opensymphony.workflow.Condition, or AbstractJIRACondition [Adds getIssue() method]
Methods:	passesCondition(Map variables, Map arguments, PropertySet) : bool
variables:	entry WorkflowEntry context WorkflowContext actionId Integer currentSteps Collection store WorkflowStore descriptor WorkflowDescriptor
arguments:	The values from the XML workflow definition
PropertySet:	The persistent state of the workflow. I think making changes here alters the workflow's state.

could do with further detail or moveing the more indepth detail of workflow functions out into an implementation section.

Workflow Function Plug-in

Note: see OSWorkflow book page 82 about testing

Note: configuration may be stored in a property set

Existing Plug-ins

JIRA has some Workflow Function Plug-ins that are already included in the standard distribution.

Set field Value on Transition

Format

```
<function type="class">
  <arg name="class.name">
    ...function.issue.UpdateIssueStatusFunction</arg>
  <arg name="field name">value</arg>
  <arg name="field name">value</arg>
  ...
</function>
```

Description

Custom fields cannot be set this way and the values they are set to are static.

BeanShell

Format

Bean shell code can be embedded in a workflow definition using the following format.

```
<function type="beanshell">
  <arg name="script">
    ...
  </arg>
</function>
```

Example

Setting a field of an Issue.

```
<function type="beanshell">
  <arg name="script">
    Object issue = transientVars.get("issue");
    issue.set("assignee",issue.getString("developer"));
  </arg>
</function>
```

Bespoke Function Plug-ins

The factory can be based on *WorkflowPluginFactory* or *WorkflowNoInputPluginFactory* depending on if it has a configuration page or not.

The function should be based on *AbstractJiraFunctionProvider*

Plug-in XML description file

The normal headers, footers and plugin-info sections are used. The section specific to Workflow Function Plug-ins is *workflow-function*. The format is:

```
<workflow-function
  key    ="unique id"
  name   ="human readable name"
  class  ="full class path to Factory implementation">
  <description>human readable description</description>
  <function-class>full class path to Function implementation
                                                                </function-class>

  <orderable>[true/false]</orderable>
  <unique>[true/false]</unique>
```

Sample Tracking: Technical Specification

```
<deletable>[true/false]</deletable>
<default>[true/false]</default>
<resource
    type      ="velocity"
    name      ="view"
    location="templates/view template file.vm" />
<resource
    type      ="velocity"
    name      ="input-parameters"
    location="templates/input template file.vm" />
<resource
    type="velocity"
    name="edit-parameters"
    location="templates/edit template file.vm" />
</workflow-function>
```

Templates

Three templates are used, their names are listed in the JavaDoc of AbstractWorkflowPluginFactory. They are:

- View - used to for the line in a transitions definition that shows the post-function. It should render to a line of text that summarizes the action the function will perform
- Input-parameters - displayed when the function is added to a workflow
- Edit-parameters - displayed if the edit link next to the 'view' is clicked

Input-parameters and Edit-parameters are very similar, differing only in that there are no previously configured values to display in Input-parameters.

View

Any parameters/settings should be put into the velocity environment by the getVelocityParamsForView method of the WorkflowPluginFactory. The method is passed an AbstractDescriptor which can be cast to a FunctionalDescriptor and used to access the configuration for that instance.

Input-Parameters

The template can assume that it is within a table.

Any parameters/settings should be put into the velocity environment by the getVelocityParamsForInput method of the WorkflowPluginFactory. As no configuration exists yet there is no Descriptor.

Edit-Parameters

The template can assume that it is within a table.

Any parameters/settings should be put into the velocity environment by the getVelocityParamsForEdit method of the WorkflowPluginFactory. The method is passed an AbstractDescriptor which can be cast to a FunctionalDescriptor and used to access the configuration for that instance. All of the parameters have to be copied explicitly from the Descriptor to the velocity params.

FunctionDescriptor

The configuration for each instance is stored in the workflow as a series of 'arg' tags within the function definition. The names and values of these are set by the 'getDescriptorParams' method of WorkflowPluginFactory. The 'getDescriptorParams' method takes a map of parameters from the POST request that was made from input-parameters or edit-parameters. It returns a second map that will be used to populate the arg tags.

The arg tags are then read in and converted into a FunctionDescriptor that is passed into 'view', 'edit-parameters' and a map of arg tag names to values is passed to the execute method of the 'FunctionProvider' class.

To access the arg values the method .getArgs() is used first and then .get("name") on the object returned to get the value.

Automatic Workflow Functions

Format

```
<action id='...' name='...' auto='TRUE'>
```

Description

The action is carried out as soon as the step is reached. Any results must be unconditional.

View Form on Transition

Format

```
<action id='...' name='...' view='fieldscreen'>
  <Meta name='jira.fieldscreen.id'>screen id</Meta>
  ...
```

JIRA GUI Modifications

The View Issue Page

JSPs

The View Issue page action is handled by the ViewIssue class, which handles URLs beginning with 'browse'. The view is generated by secure/views/issue/ViewIssue.jsp. ViewIssue.jsp includes several other JSPs:

- /includes/errormessages.jsp
- /includes/panels/issue_headertable.jsp
- /includes/panels/issue/view_timetracking.jsp
- /includes/panels/issue/view_attachments.jsp
- /includes/panels/issue/view_linking.jsp

Includes:

- /includes/panels/issue/linking/outward.jsp
- /includes/panels/issue/linking/inward.jsp
- /includes/panels/issue/view_trackbacks.jsp
- /includes/panels/issue/view_customfields.jsp

Includes FieldScreenRenderTab objects that include the custom fields

(All of the tabs are included in the HTML. Javascript is the used to display only the currently active tab. The names of the tabs cannot contain HTML.)

- /includes/panels/issue/view_subtaskissues.jsp
- /includes/panels/issue_descriptiontable.jsp
- /includes/trackback_rdf.jsp
- /includes/panels/issue/actions.jsp

Includes The Tab Panels

(Only the current Tab Panel is included in the HTML)

Some of the JSPs are only optionally included.

Pluggable Issue Operation

The list of operations only appears on the View Issue screen and so could be used to add HTML/Javascript to the view. The 'showOperation' method is passed an Issue object and so the plug-in's use could be limited to set projects.

Web Fragments

In version 4.0; there is no way to add a fragment to just the View Issue page.

IssueTabPanel (only rendered when open)

IssueTabPanels are only rendered when they are the active tab and so are not a reliable way to add to the view.

There is a lack of id's in the surrounding tags and so any hiding of the tab would have to be done using an anchor added by the 'label' and looking for the wrapping list item tags.

Pluggable Footer

While pluggable footer can appear on every page it doesn't necessarily have to. The footer's get HTML methods, and the velocity template's context, are passed a HttpServletRequest object that can be used to determine what page was requested. When the issue view is being returned webwork.view_uri is '/secure/views/issue/viewissue.jsp' and webwork.request_uri is '/jira/secure/ViewIssue.jsps'

Every Page

Announcement Banner

The announcement banner is a built-in feature. A block of text, or HTML, can be configured by an administrator via the existing GUI to be added to every page.

Advantages

- Currently not being used
- Added to most pages

Disadvantages

- Separate from the plug-in
- Confusing if someone tries to use the banner without knowing about the JS

JSPs

Each web page begins with:

includes/decorators/header.jsp

The header unconditionally includes the following files in each page

/includes/decorators/uacompatibility.jsp

/includes/decorators/styleheettag.jsp

/includes/decorators/bodytop.jsp includes:

- PluggableTopNavigation objects from their TopNavigationModuleDescriptors
- The Announcement Banner

/includes/decorators/footer.jsp includes:

- PluggableFooter objects from their FooterModuleDescriptors

Pluggable Footer

The pluggable footer is inserted into every page. Implementations need to be able to handle being used on all types of pages, including where the user is not logged in. This can be as simple as testing some of the request's properties to decide if it should generate any HTML.

Pluggable Top Navigation

Similar to the Pluggable Footer apart from where it is inserted and its original purpose.

Required JIRA Customizations

- Sub-Task complete email parent assignee
- Comment required to deprecate
- Email on entering state
- Wait for sub tasks to finish
- Wait for batch available
- Flurp creation of JIRA Tasks
 - Mini functional specification
- Compound fields
- Increment Assembly Versions
- Values from the database
 - One time
 - Updating
 - Chart
 - Searches
- set is_draft in GLK
- set deprecated in GLK
- set ?batch? in GLK
- Auto running VAPOR
- Run autotasker
- Update number of tasks field
- hiding sub-tasks other than in Lab processing and closure
- Sub task naming
- Using BAC id's to locate issues
- Import of Legacy data
- **Prevention of duplicate entries**

Customizations by Workflow Step

Step	Component	Notes
Received Sample	Import Issue	
	DB Fields	+Search of DB Fields
	Quick Search	
	Velocity Fields	
	DB Write	
Lab Processing	Subset of Sub-tasks	
Assemble	Call External Program	
	External Program moves workflow	
	Velocity Field	# of assemblies
Review Assembly	Call External Program	Autotasker2
	Results of call in field	Autotasker2
	Velocity field	Appending the results field to the # of tasks field
Annotate	'Requires Annotation' test	
Review Annotation	'Requires Annotation' test	
Validate	Call External Program	?
Collaborator Review	Set Wait Date	
	Wait for Date	
	Wait for Batch	
Deliver Data		
Submitted to Genbank		
Sample Published		
Close Sample	Subset of Sub-Tasks	
Unresolved		
Deprecated	DB Write	

Customizations by Field

Name	Format	Components required
Sample Id:	<Database>_<Collection Code>_<BAC Id>	Import
Summary:	<Sample Id>	Import
Priority:	undefined	
Due Date:	undefined	
Assignee:	undefined, possibly linked to database or collection	
Reporter:	<current user>	
Description:	FLURP: Screen2 - Collection Lot Description?	
Charge Code:	undefined	

Sample Tracking: Technical Specification

Sample Arrival Date:	undefined, SPREADSHEET:Date Sent to JCVI?	
Database:	FLURP: Screen2 - Database Name	Import
Collection Code:	SPREADSHEET:Blinded ID Number(NIGSP_DD2_00002) FLURP: Screen2 - Collection Code	Import
BAC Id:	SPREADSHEET:Blinded ID Number(NIGSP_DD2_00002)	Import
Blinded Number:	<Project>_<Collection Code>_<BAC Id> SPREADSHEET:Blinded ID Number (NIGSP_DD2_00002)	Velocity Field?
Computed Subtype:	SPREADSHEET:Organism Name (Influenza A virus (A/Sydney/DD2-01/2010(H3N2)) FLURP: Screen2 - [Generate Subtypes]	DB Field
CC Users:	undefined, based on collection code	
Attachment:	NONE	
Genbank Submission Date:	NONE	
Path to Assembly Files:	NONE	
Assembly Quality Status:	NONE	
VGD Loaded:	NONE	
VIGOR Run:	NONE	
Sample Tracking Bin:	NONE	

Requires synchronizing, Sub-part of field only, Hidden

Which of the JIRA fields should be made read only, e.g. blinded number not being edited directly, only via BAC id, collection code etc

Components

DB Synchronizer

Overview

Implementation

Custom Field Type

Sub-components:

- synchronizer
- formatter
- initial value setting
- interceptors
 - create
 - search
 - view
- Working with Search, Charts etc

Workflow SQL Runner

Overview

Runs a piece of SQL, mostly for updates but it could possibly populate a field

Implementation

Workflow Function Plug-in

Workflow Set Field from File

Overview

Reads a file to populate a field. The file's path will need to be constructed using variable substitution or velocity. Velocity can already do this, but only within the document root. Just set the document root to the files ystem root?

Implementation

Workflow Function Plug-in

Concatenate multiple files

Workflow Program Runner

Overview

Calls a program, with variable arguments, and then, optionally, sets a field or comment based on the result.

Implementation

Set <Field>

Append comment

path: see set field from file

Sample Tracking: Technical Specification

parameters: see set field from file

field = stdout

field = stderr

field = 'Message' on return code X

field = 'Message' on default return code

```
import java.io.*;

public class Foo {
    public static void main(String args[]) {
        String s;

        try {
            // Over Java 1.5
            Process oProcess = new ProcessBuilder("cmd", "/c", "dir", "/?").start();

            // Read output from external program
            BufferedReader stdout = new BufferedReader(new
                InputStreamReader(oProcess.getInputStream()));
            BufferedReader stderr = new BufferedReader(new
                InputStreamReader(oProcess.getErrorStream()));

            while ((s = stdout.readLine()) != null) System.out.println(s);
            while ((s = stderr.readLine()) != null) System.err.println(s);

            // Print return value
            System.out.println("Exit Code: " + oProcess.exitValue());
            System.exit(oProcess.exitValue());

        } catch (IOException e) { // Err processing
            System.err.println("Err !! \n" + e.getMessage());
            System.exit(-1);
        }
    }
}
```

Workflow - block until batch is ready

Overview

Samples shouldn't move out of collaborator review until the whole batch is ready. There will probably also need to be a search that can identify the members of the batch.

Possible Implementations

- Implement condition (So the option doesn't appear for the samples that aren't ready)
- Use a search as the test, +range functions
- Use a second field to record if an issue has reached Collaborator Review; then the search can be on find #issues where
batch=<the sample's batch ID> and <Collaborator Review step flag> != true
- I don't know if it's possible to do such a search from within a condition

Issue creator

Overview

Creates new issuses when they are added to flurp
To include uploading legacy issues
Handle multiple creation attempts without creating multiple issues

Implementation

- JIRA CLI
- FLURP.RURP

Sub-Task Creator Permission

Overview

hides create sub-task based on a permission, separate to Issue create, that can be used in workflows

Possible Implementations

- action interceptor
- interceptors on the show sub-tasks tests
- modifying the issue to look like a sub-task, causing the show sub-tasks tests to fail

Sub-Task Summary Field

Overview

Summarizes the state of an Issue's sub-tasks and provides a new search that uses this information.
Possible searches clauses are:

- has X sub-tasks +range functions¹⁰
- has X sub-tasks with step in [step1, step2,...]
- has X sub-tasks with <sub-task type>.<field> = N

Possibly it should be implemented as a custom JQL function instead of a field

Implementation

Custom Field

- configuration interface

Velocity Field

Overview

Combines with DB plug in to produce the compound fields

Implementation

Custom Field

- configuration interface

Hidden JS Field / Footer

Overview

Provides a method of adding Javascript to various parts of the GUI.

¹⁰ Range functions are; value is less than X, is greater than X, is between X and Y

Implementation

Using a hidden Custom Field; edit and view can have separate content

Using a pluggable footer; No selection of when it is and isn't used provided directly but could be built on top of the information from the HTTP request. It would allow access to more screens of the GUI.

An administrator GUI to allow configuration would be needed.

An XML wrapped import/export for updating from a test system would be good.

Testing

- A full list of functions would be good
- Use Cases (for later user docs)

Implementation log?

- Somewhere to keep the notes on what didn't work
- How to setup and develop for JIRA

Reports

Many of the reports share SQL with each other. The first large group are those that use the Extent, CTM_Reference, CTM_Reference Status and BAC tables.

Extent ctm_reference ctm_reference_status and bac Tables

They are all based on the SQL in this template:

```
select <selected columns>
from Extent e, ctm_reference r, ctm_reference_status s, track..bac b
where
  r.ctm_reference_status_id = s.ctm_reference_status_id and
  convert (numeric,e.ref_id)=b.id and
  convert(numeric,r.value)=e.Extent_id
  [AND s.name like "<sname value>"]
  [AND b.lib_name like "<lib_name value>"]

[order by <order by column>]
```

The only differences between all of these queries are summarized below:

Report Name	selected columns	s.name	b.lib_name	order by column
closure_q_collection	e.ref_id	'Coinfection/Mixed'	"DW%"	-
single_sample_status	e.ref_id, b.lib_name, s.name	e.ref_id="29725"		
multi_closure_q	e.ref_id, b.lib_name, b.lib_desc	in ('Primer Design', 'Hold', 'Temp Hold', 'Coinfection/Mixed')		
one_person_q	e.ref_id, b.lib_name, b.lib_desc	'Closure KD'		
closure_q	e.ref_id, b.lib_name, b.lib_desc	'DraftSubmission'		b.lib_name
closure_q2	e.ref_id	'Pending'		
Closure_stat_specific	e.ref_id, b.lib_name, s.name		"DW09%63"	
FL_status_sql SDI0708_status_sql	e.ref_id, b.lib_name, s.name		in ("DW0900136",...)	s.name
closure_status	e.ref_id, b.lib_name, s.name		"DW%"	
closure_status2	e.ref_id,		"DW%"	
closure_status_count	count(*) ¹¹		"DW%"	

Extent, ctm_reference and ctm_reference_history Tables

others are based on:

```
select distinct e.ref_id
from Extent e, ctm_reference cr, ctm_reference_history h
where
  convert(numeric,cr.value)=e.Extent_id and
  cr.ctm_reference_id=h.ctm_reference_id and
  h.ctm_reference_status_id=<status id> and
  h.date_in > "<start date>" and
  h.date_in < "<end date>"
```

Report Name	status id	status name
validation_q	66	

¹¹ group by s.name

Sample Tracking: Technical Specification

collection_val_giv3	86	Collection Validation
collection_val_swiv	68	
DW09_pending_giv	81	Pending
validation_giv	23	Validation
validation_giv3	25	
validation_swiv	66	

The start date for all of the queries was "Aug 31 2009" and the end date was always "Oct 01 2009".

ctm_reference, ctm_reference_history, ctm_reference_status and two Extent Tables

```
select collection.ref_id as collection, sample.ref_id as sample, h.date_in
from Extent collection, Extent sample, ctm_reference r, ctm_reference_history h, ctm_reference_status s
where
    h.ctm_reference_status_id=s.ctm_reference_status_id and
    r.ctm_reference_id=h.ctm_reference_id and
    s.name="<name value>" and
    sample.Extent_id=CONVERT(NUMERIC,r.value) and
    sample.parent_id=collection.Extent_id and
    collection.ref_id NOT IN ('XX') and
    [h.date_in > '<start date>' and
    h.date_in < '<end date>' ]
order by h.date_in
```

Report Name	status name	start date	end date
closure_q_date	Validation	-	-
submission_q	Published	Dec 31 2007	Jan 1 2009
validation_q2	Validation	Jan 31 2009	Feb 28 2009

gel_query

gel_query is one of two unique queries:

```
select r.ctm_reference_id, l.bac_id, q.seq_name, q.ed_ln
from track..library l, Extent e, track..sample s, ctm_reference r, sequence q
where
    l.lib_id=s.lib_id and
    q.seq_name=s.seq_name and
    convert (varchar(36),l.bac_id)=e.ref_id and
    convert (varchar(30),e.Extent_id)=r.value and
    s.gel_id between 1040499 and 1040500
order by l.bac_id
```

ctm_reference_id	bac_id	seq_name	ed_ln
------------------	--------	----------	-------

task_count_year

gel_query is one of two unique queries:

```
select ctm_task_type_id, creation_date, sample
from ctm_task
where
    creation_date > 'Jan 01 2010' and
    creation_date < 'May 11 2010' and
    ctm_task_type_id = 1
```

Annotation

The typical distribution of responsibility, and the exceptional case for Influenza Samples

	Normally Responsible	Responsible with Influenza Samples
Generate Annotation	The Annotation Group	Skipped
Review Annotation		
Validation		The Closure Group

For normal annotation:

- The db needs to be loaded (NGS Task?)
- The annotation pipeline is run. The pipeline uses tools such as
 - Autotasker2
 - Vigor
- The validation includes preparing submission files for Gen-bank and trace archives.

The person responsible for this process varies from project to project. There is a command 'getBA' which should convert a db name into the name of the BA for the project.