iTop Advanced Usage Documentation

- The documentation about integration explains how to import/export and synchronize data in order to integrate iTop with other applications
- The customization documentation explains how to extend (or modify) the data model of the application

Old Documentation

The following documentation applies to iTop versions 1.0 up to 1.2.1 only

- Toolkit for modifying the data model
- How to create your own module

Advanced topics

A CMDB is rarely a stand-alone application. In order to integrate iTop with the rest of your infrastructure, three powerful mechanisms are available:

- An Export page than can be used either as a web service or from the command line. The page allows to export in XML, CSV or HTML format any set of iTop objects that can be described by an OQL query.
- A scripted Import page available as a web service or from the command line to import or update iTop objects from a CSV file
- A Data Synchronization engine that enables iTop to federate data from various sources

All these services work on virtually any object in iTop: Devices, Contacts, Tickets, User Accounts…

What is the difference between CSV Import and Data Synchronization ?

Data synchronization is meant to import data into iTop, from another system, in a recurring manner. It can be run from the command line or from a web service, but not interactively. Data Synchronization is optimized for large volume of data that do not change very often. For example you may synchronize 10,000 contacts from an LDAP server, in iTop, once per day. Everyday probably only a small percentage of the users' records will be modified. This is efficiently handled by iTop.

When synchronizing data, iTop keeps track of the relationship between the iTop object and the source of the data. Therefore it is possible to prevent the users from modifying the synchronized objects (partially or totally) in iTop and to tell them where the data comes from. This is useful for "federating" several sources of data in iTop.

The CSV import (which can run interactively or from the command line) is more targeted towards "one shot" import. It can be used from a script (using the command line interface or the webservice) or interactively. When performing a CSV import, iTop does not record information about the source of the data. Once the data have been loaded into iTop, the objects can be modified by the authorized users, without any reference to the original source.

To summarize:

CSV import is good for:

- importing initial data in iTop
- performing bulk transformations on the data (sometimes it's easier to export / modify in Excel / reimport than to edit the objects directly in iTop)

Synchronization is good for:

- federating data between different systems in iTop
- importing data via some scheduled mechanism
- preventing users from modifying the imported data

REST/CLI services

The import, export and data synchronization pages can be run either as REST web services or from the *Command-Line Interface* (CLI). Since the CLI mode can only be used by scripts running on the iTop server itself it is considered as safer and generally runs with less limitations compared to the web pages (on most system the CLI mode has its own php.ini configuration). For example:

- CLI mode can benefit from longer timeout or no timeout at all (useful for running big imports)
- The `memory_limit` setting may be set to a bigger value for PHP scripts running in CLI mode.

-

In CLI mode, the arguments to iTop pages are always given with the prefix -- (two consecutive hyphens).

Example (on Windows):

```
php.exe -q c:\inetpub\itop\synchro\synchro_exec.php --auth_user=john
--auth_pwd=trust,no1
```

Same example (on Linux):

```
php -q /var/www/itop/synchro/synchro_exec.php --auth_user=john
--auth_pwd=trust,no1
```

or, even better, using a *parameters file*:

```
php -q /var/www/itop/synchro/synchro_exec.php
--param_file=/etc/itop/params.foo
```

On some Linux systems the PHP command-line interface must be installed as a separate package (called `php-cli` or `php5-cli`). To check if the PHP CLI is installed (and available in the path) on your system simply type `which php` from the command prompt.

Using a parameters file

For security reasons it is always better to avoid passing credentials (user names and passwords) on the command line (the command lines corresponding to the processes running on the system are generally visible to all users logged-in). iTop offers the alternative to pass all command-line parameters inside a file, called the “parameters file” via the argument param_file . This argument can be used with most of the REST/CLI web services, it must contain the path to a parameters file.

Make sure that the parameters file is readable by the process that will run the PHP page, and that it cannot be accessed through the web server.

A parameters file contains key/value pairs and always uses the same format. It can be commented: any character found after # on a given line will be ignored

When a parameter is specified both in the param file and as a command-line argument, the value given on the command line has precedence.

Example:

params

```
# This is a parameter file
#
# If a parameter is given both in the file and in the arguments,
# then the value given as argument is retained
#

# Authentication
auth_user = qwertyuiop
auth_pwd = ded!catedL0g1n

# My web service
size_min = 20 # Megabytes
time_limit = 40 # Minutes
```

List of REST/CLI services

- cron.php: the heartbeat of iTop, enables some features like asynchronous emails
- synchro_exec.php: trigger the synchronization of a list of data sources
- synchro_import.php: in one single operation, import data and trigger the synchronization process
- import.php: import data from CSV files
- export.php: export data in various formats

REST/JSON services

This feature will be available with iTop 2.0.1

REST/JSON services are generic services. The APIs are low-level operations (search objects, create/update/delete objects) that will be the building bricks allowing any kind of integration.

Furthermore, a custom module can provide higher level operations that will be delivered through this same entry point.

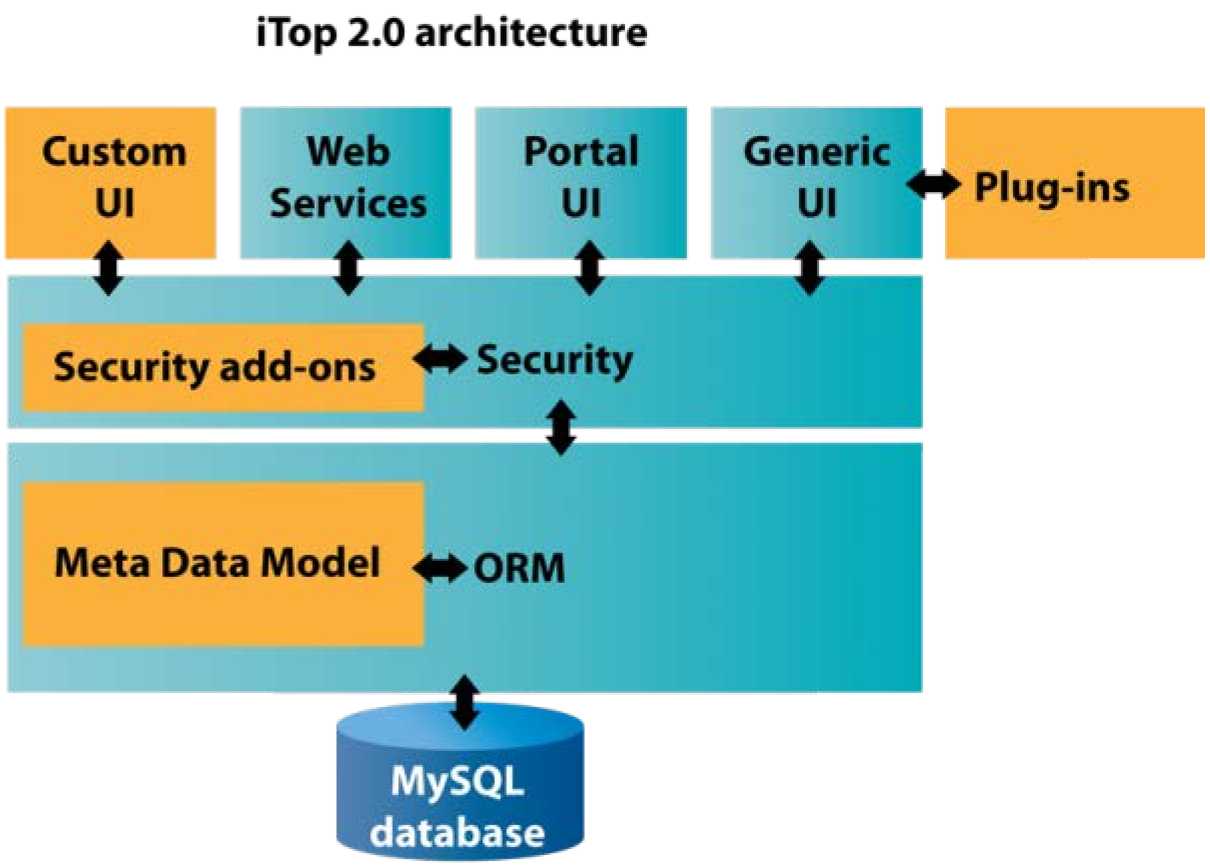
iTop Customization

This document applies to iTop version 2.0. For customizing previous versions of iTop refer to the old toolkit. The 2.0 toolkit is available here: iTopDataModelToolkit-2.0.zip

iTop is built on top of an ORM (Object Relational Mapper) abstraction layer that relies on the definition of a “Meta Data Model” made of PHP classes.

Starting with iTop 2.0 the Meta Data Model can be described in XML which is then “compiled” into PHP classes during the setup of the application.

The iTop architecture can be depicted as the schema below:



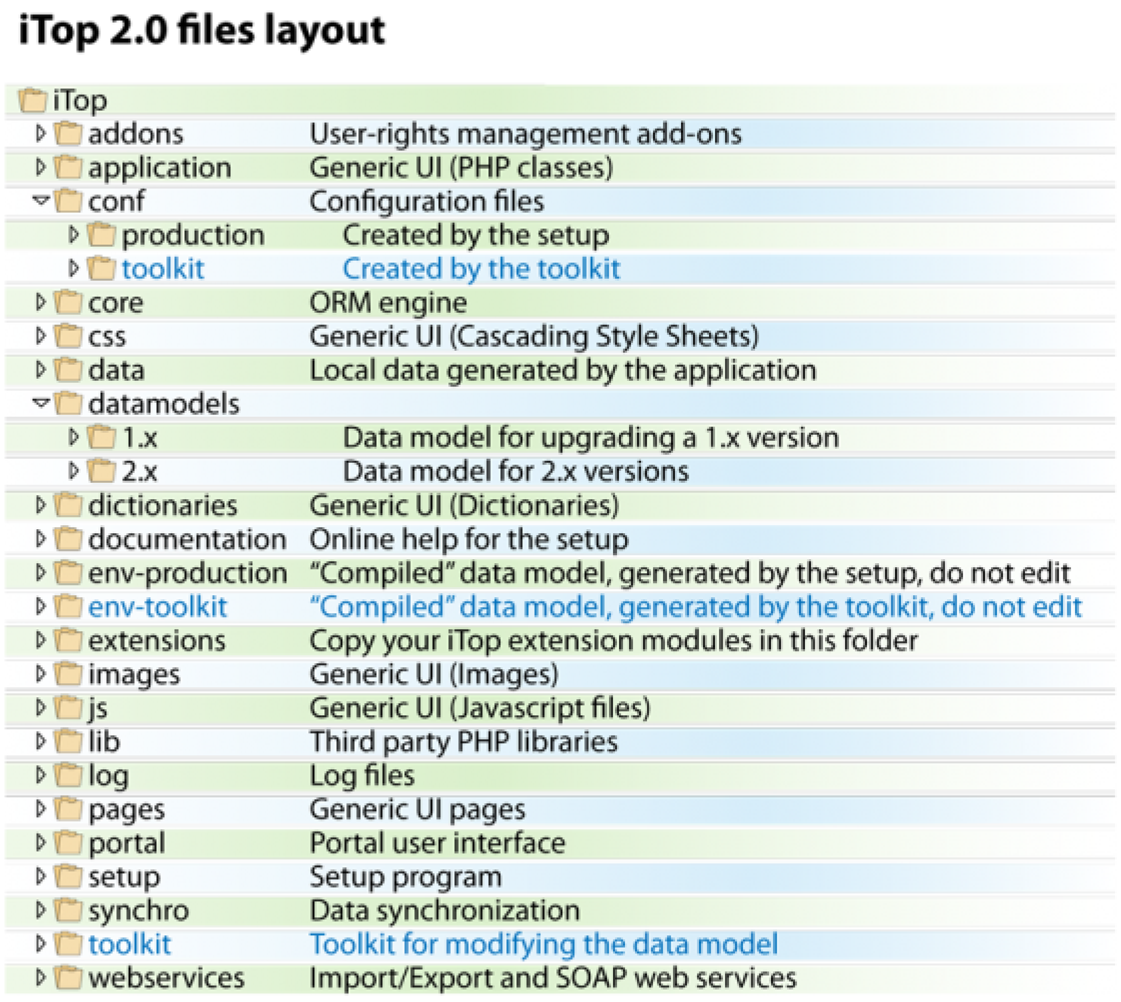
The orange boxes on the schema above are the parts of the application that can be customized.

Type of customization	Mean	Documentation	Intended audience
Extending or modifying the meta data model	XML files	[XML reference]()	iTop consultants, ITIL specialists
Extending the generic user-interface via a plug-in	PHP files	[Extensions API]()	PHP developers
Creating new user interface pages to implement new web services, specialized exports or a specific task oriented user interface	PHP files	ORM API	PHP developers
Creating your own security scheme	PHP files	User Rights API	PHP developers

Starting with iTop 2.0, the modification of the meta data model can be done by writing a simple XML file that will contain only the differences with the standard data model. By doing so, your modifications are kept in a separate file, that will remain applicable to the next version of iTop.Thus preserving your customizations in case of upgrade.

Understanding the iTop file structure

The layout of the iTop files and folders can be depicted as below:



iTop 2.0 files layout

iTop	
addons	User-rights management add-ons
application	Generic UI (PHP classes)
conf	Configuration files
production	Created by the setup
toolkit	Created by the toolkit
core	ORM engine
css	Generic UI (Cascading Style Sheets)
data	Local data generated by the application
datamodels	
1.x	Data model for upgrading a 1.x version
2.x	Data model for 2.x versions
dictionaries	Generic UI (Dictionaries)
documentation	Online help for the setup
env-production	"Compiled" data model, generated by the setup, do not edit
env-toolkit	"Compiled" data model, generated by the toolkit, do not edit
extensions	Copy your iTop extension modules in this folder
images	Generic UI (Images)
js	Generic UI (Javascript files)
lib	Third party PHP libraries
log	Log files
pages	Generic UI pages
portal	Portal user interface
setup	Setup program
synchro	Data synchronization
toolkit	Toolkit for modifying the data model
webservices	Import/Export and SOAP web services

The following folders have a special usage in iTop:

- **conf**: contains the configuration files, with one sub-folder per environment (see Environments below).
- **data**: contains application generated data, like the image for the object life-cycle (if graphviz is available on the system)
- **datamodels**: contains the meta data model definitions, with one sub-folder per major version of iTop.
- **env-xxxx**: these folders (one per environment) contain the "compiled " data model. The env-production folder is completely re-created each time the application is re-installed

or upgraded. If you edit its content, be aware that your modifications *will be lost* upon re-install or upgrade.

- **extensions**: this folder is the place where to copy extensions modules that are not part of the standard distribution of iTop.
- **log**: contains the log files of the application: setup.log and error.log

All the other folders should be considered as part of the source code of the application and should generally not be modified. The application never writes to these folders, they can be marked as read-only for the web server process.

Environments

A new concept introduced by iTop 2.0 is the notion of “environment”. An environment is an instance of iTop running the same code base but with potentially its own data model and configuration file. An environment is made of:

- A configuration file stored in conf/name_of_the_environment/config-itop.php.
- A data model runtime, stored in env-name_of_the_environment

Two different environments can operate on the same database if their configuration files says so.

The toolkit automatically creates a separate environment (named toolkit) in order to compile the XML data model and test its consistency without affecting the “production” environment. When the changes are Ok, you can instruct the toolkit to apply the validated changes to the “production” environment.

Extension Modules

The basic unit of data model definition in iTop is called a module.

A module groups together all the files needed to deliver a given feature: data model definitions in XML, PHP classes, Javascript and CSS files, PHP pages, images, etc… A module contains at least one file: the module definition file, always named module.name_of_the_module.php.

Though you can always patch the source code, the best way to customize iTop consists in writing your own module. This creates a clean packaging of the customization and allow an easy (re)installation for your deployment or in case of upgrade.

PHP versus XML data model definitions

In iTop versions 1.x, the data model definition was written as a plain PHP classes. iTop version 2.0 support both PHP and XML definitions for the data model.

XML definitions have one major advantage over PHP definitions: the XML definition in one module can alter the data model defined in another module. For example, it is possible to create an extension module that will - when installed - add an attribute to the standard class “Server”, which is defined in the standard iTop data model. The extension module does not need to replace the whole “itop-config-management” module (where the Server class is defined), it can just alter the definition of a classe defines elsewhere.

In order to achieve the same effect in PHP, the only solution consists in cloning the whole itop-config-mgmt module and replacing it with your own patched version. When a new version of the module is released, you have to redo this diff & patch work again to produce a new version of your own module. Since an XML definition is directly defined as a difference, the upgrade is automatic.

Content of a module

If *my-module* is the name of your module, the module folder will contain the following files:

File Name	Description
module.*my-module*.php	The module definition file. Mandatory. Contains the description of the module (name version, dependencies on other modules, etc.) and its components.
datamodel.*my-module*.xml	Data Model in XML. Upon installation the “compilation” will produce the model.xxxxx.php file based on the XML definitions. The XML file can contain the definition of classes, menus and profiles
model.*my-module*.php	If you choose to define the data model directly in PHP, then this file is the place to put such definitions
main.*my-module*.php	PHP code and utilities. For some modules that contain a lot of PHP code, it is easier to extract the PHP code and edit it in this separate file than letting the code embedded in the XML.
images	It is a good practice to store the images (classes icons, etc…) in their own sub folder

The module names starting with *itop-* are reserved for offical modules from the iTop package. To avoid naming conflicts with other extensions it is recommended to name your custom modules with a name starting with the name of your company. For example the custom modules developed by Combodo are named *combodo-xxxx*.

Creating a module

Fill the form below and click on “Generate” to generate an empty module as the starting point for your customization:

Module Name:	sample-modul	(must be unique. Names starting with "itop-" and "combodo-" are reserved for use by Combodo)
Module Label:	Sample Modul	(Displayed during the setup)
Module Version:	1.0.0	
Category:	business	
Dependencies:		(comma separated list of module names/versions)

Generate !

Installing the Toolkit

- Download the toolkit zip file: iTopDataModelToolkit-2.0.zip
- Expand the content of the zip file to create a directory “toolkit” at the root of the your developement iTop instance.
- Point your browser to **http://<your_itop>/toolkit**

Development Workflow

1. Create an empty module
2. Install a development instance of iTop, including your empty module in the “extensions” folder
3. Install the toolkit on your developement instance
4. Edit your extension module and validate it with the toolkit
5. Apply the changes made to your extension module to the “production” environment
6. Test your module with some sample data. In case of troubles, fix them by iterating at point 4.

XML files must be “compiled” to PHP files each time you modify them, this is the job of the toolkit. However all other files in your module are simply copied from the "extension/your_module" folder to "env-production/your_module" folder. To speed-up the debugging on Linux systems you can replace these files by a symbolic link to their actual source file; Thus any modification to the source is immediately effective in iTop, you just have to hit the refresh button of the browser.

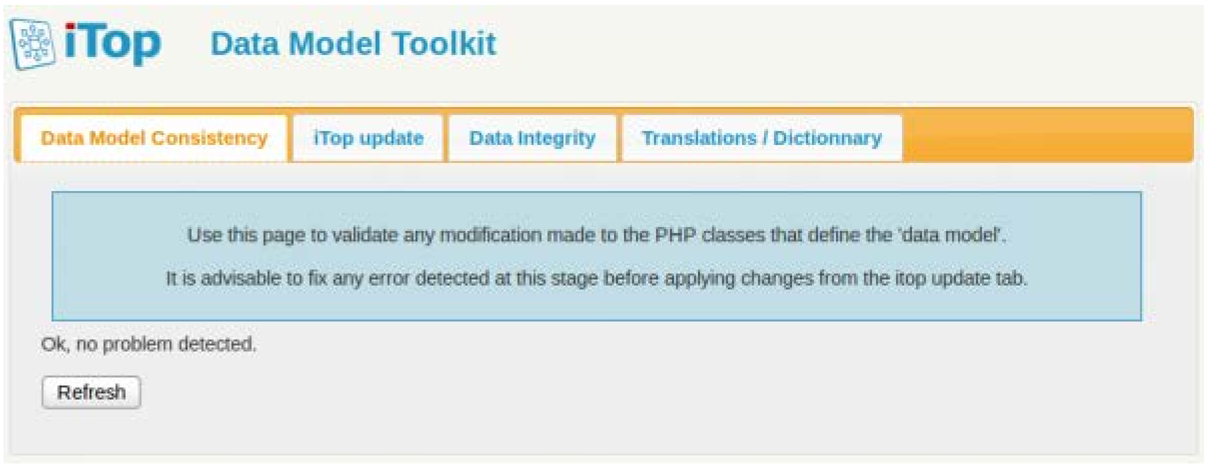
When your extension is completed you can deploy it on your production system by:

1. Copying the folder containing your extension module to the "extensions" folder on the production system
2. Marking the configuration file as read/write
3. Running the setup again and selecting your module in the list of "extensions" at the end of the interactive setup

Using the Toolkit

Once the toolkit is installed, point your browser to: **http://<your_itop>/toolkit**.

The first tab performs some basic consistency checks and validation of the data model definition. You can use the "Refresh" button to perform the validation again each time the data model definition has changed. The checks performed in this tab work on the specific "toolkit" environment and thus have no effect on the actual iTop instance that uses the "production" environment.



Use this page to validate any modification made to the PHP classes that define the 'data model'.

It is advisable to fix any error detected at this stage before applying changes from the itop update tab.

Ok, no problem detected.

The order in which the XML files are loaded is important (especially if an XML file alters the definitions given in another one). This loading order follows the "dependencies" declared in the module description file.

If you get an error like:

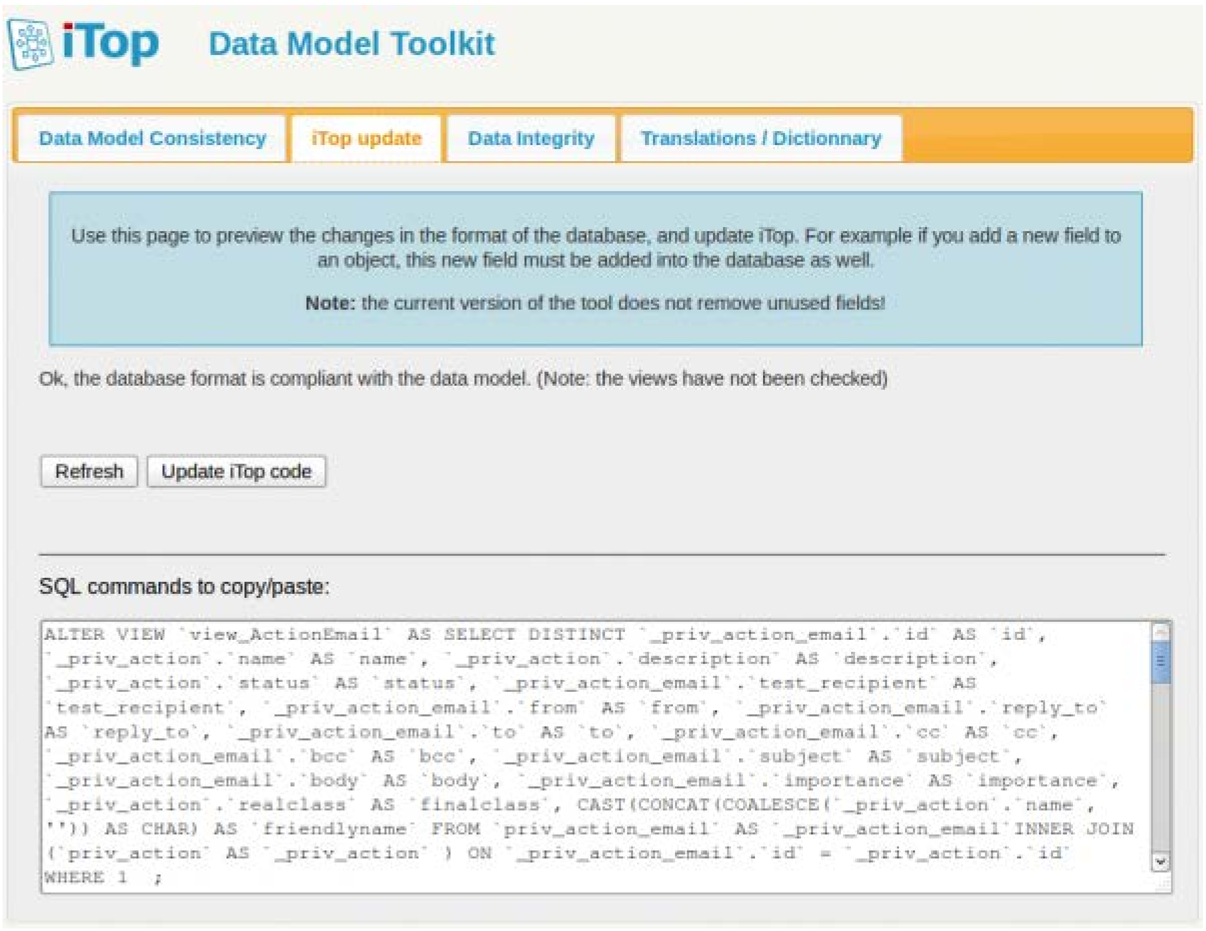
```
XML datamodel loader: could not find node for class/XXXX
```

This probably means that your module lacks a dependency on the module where the class XXXX is defined.

When the setup runs the loading order is computed and the resulting order is written in the configuration file. Be aware that the toolkit does *not* recompute this loading order. Therefore if you miss a dependency, you need to modify the module definition file to add this dependency and run the setup again in order to recompute the whole configuration file

When the first tab does not show any error, you can move to the second tab to:

- check the change to the database schema
- apply the changes of the database schema and data model definitions to the "production" iTop environment by clicking on "Udpate iTop Code"



Use this page to preview the changes in the format of the database, and update iTop. For example if you add a new field to an object, this new field must be added into the database as well.

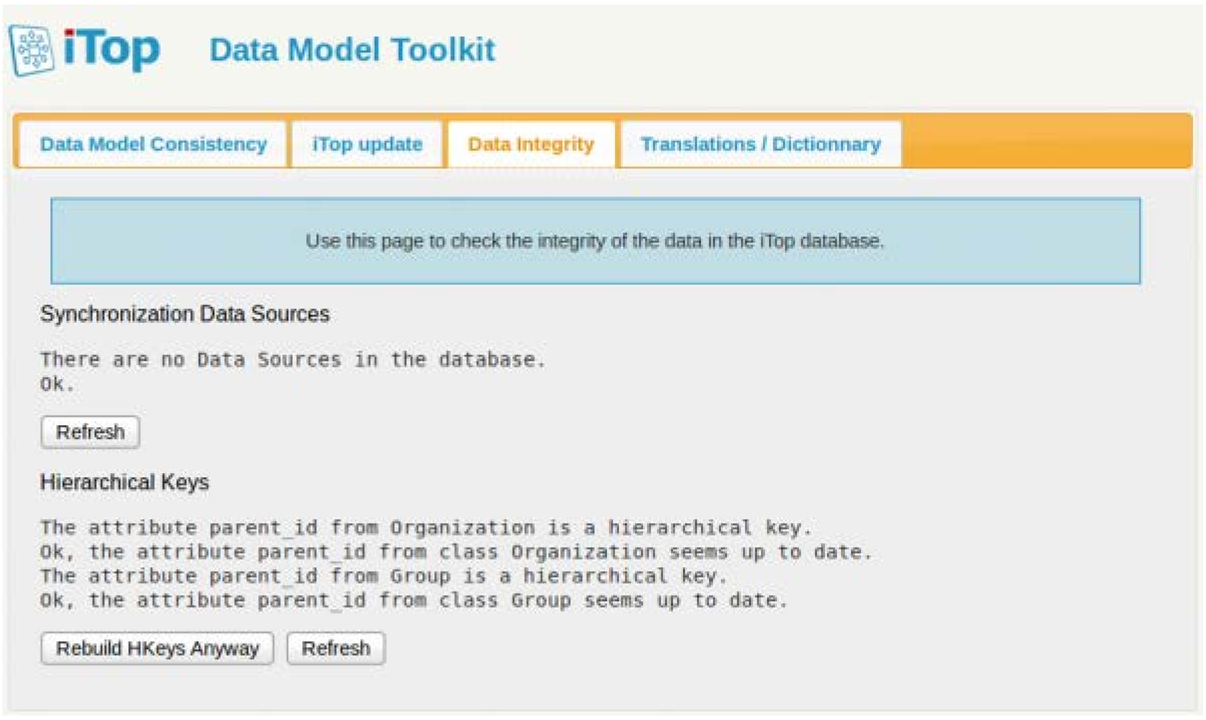
Note: the current version of the tool does not remove unused fields!

Ok, the database format is compliant with the data model. (Note: the views have not been checked)

SQL commands to copy/paste:

```
ALTER VIEW `view_ActionEmail` AS SELECT DISTINCT `_priv_action_email`.`id` AS `id`,
`_priv_action`.`name` AS `name`, `_priv_action`.`description` AS `description`,
`_priv_action`.`status` AS `status`, `_priv_action_email`.`test_recipient` AS
`test_recipient`, `_priv_action_email`.`from` AS `from`, `_priv_action_email`.`reply_to`
AS `reply_to`, `_priv_action_email`.`to` AS `to`, `_priv_action_email`.`cc` AS `cc`,
`_priv_action_email`.`bcc` AS `bcc`, `_priv_action_email`.`subject` AS `subject`,
`_priv_action_email`.`body` AS `body`, `_priv_action_email`.`importance` AS `importance`,
`_priv_action`.`realclass` AS `finalclass`, CAST(CONCAT(COALESCE(`_priv_action`.`name`,
'')) AS CHAR) AS `friendlyname` FROM `priv_action_email` AS `_priv_action_email`INNER JOIN
(`priv_action` AS `_priv_action` ) ON `_priv_action_email`.`id` = `_priv_action`.`id`
WHERE 1  ;
```

The third tab of the toolkit can be used to update the Data Synchronization sources (if there are any that was impacted by the changes to the data model), and to check if there are any discrepencies in the internal data maintained for hierachical keys. This tab directly operates on the "production" environment.



Use this page to check the integrity of the data in the iTop database.

Synchronization Data Sources

```
There are no Data Sources in the database.
Ok.
```

Hierarchical Keys

```
The attribute parent_id from Organization is a hierarchical key.
Ok, the attribute parent_id from class Organization seems up to date.
The attribute parent_id from Group is a hierarchical key.
Ok, the attribute parent_id from class Group seems up to date.
```

XML Data Model Reference

This document is the reference documentation for the format of the XML used in the datamodel.xxxx.xml files inside iTop modules. An XML data model file can be used to define:

- classes – that will generate the actual PHP classes when "compiled" using either the setup program or the toolkit
- menus – to be inserted in the application's menu on the left of the iTop pages
- profiles – to manage the access rights to the iTop objects

Principles

An XML data model contains both *initial declarations* and/or *alterations* of declarations made in other XML files.

The iTop compiler works in two steps:

1. Load the XML data model files of the installed modules. The XML are combined into one single XML definition. It is important to figure out that, starting from an empty definition, iTop will merge each XML one by one.
2. Interpret the final combined definition.

The XML format reference described hereafter applies to the combined definition. Thus an XML may not contain all the mandatory nodes, but only those required to identify the path of an alteration and those required to (re)define items.

By default, the contents of a given XML are merged into the combined definition.

You can define alterations using the attribute _delta. This attribute specifies how a given node must be considered, including its subnodes.

_delta can take the following values:

_delta	Meaning
merge (default)	Create this node if it does not exist already. Then examine the delta in its child nodes. This is mostly relevant for the structure nodes
must_exist	Check that this node already exists. Then examine the delta in its child nodes.
define	This is the first definition of this node. No flag should be found into its child nodes.
define_if_not_exists	It the node does not already exist then define it. Ignore it otherwise.
redefine	The contents of this node must be replaced by the contents of the delta node.
delete	Delete this node. This node should not have child nodes (no contents)

A node is identified by its tag name and the attribute id. If no id is given, then the node is identified as being the FIRST node being found having the same tag name.

Another attribute is _rename_from. Use it to specify that the item (e.g. an attribute) is in fact an existing item renamed from _rename_from to id.

When the compiler encounters define/redefine, it considers that the child nodes are pure content definition. As a consequence, any attribute _delta or _rename_from will be ignored.

XML general structure

Use the triangle arrow on the left of an item to expand/collapse its definition.

Tag	Usage	Description
<itop_design xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0">	mandatory	Structural node
<classes>	mandatory	Declared classes
<class id="name">	zero or more	Declaration of class
<parent>cmdbAbstractObject<parent>	mandatory	Parent class
<properties>	mandatory	

<is_link>1<is_link>	optional	Differentiate classes used for linking other classes together. Set to 1 for a linking class.
<comment>	mandatory	PHP comments added into the compile file, before the declaration of the class
<category>bizmodel,searchable,structure<category>	optional	Usages that will be made of this class.
<abstract>false<abstract>	mandatory	Abstract classes can not be instantiated.
<key_type>autoincrement<key_type>	mandatory	Always set to "autoincrement"
<db_table>name<db_table>	mandatory	Name of the MySQL table used for this

		class. The name given here will be automatically predended with the suffix provided at installation if any.
<db_key_field>id<db_key_field>	mandatory	Always setting the identifier field to "id" is fine
<db_final_class_field>finalclass<db_final_class_field>	mandatory	If the class is on top of a hierarchy of classes, then you must define which MySQL column will be used for keeping track of the real class of object instances. Setting this to "finalclass" is fine.
<naming>	mandator	Define how the friendly name

	y	will be computed
<format>%1$s<format>	mandatory	This printf formatting expression defines how the attributes will be put together. When specified as an empty string, the friendly name will be rendered as a concatenation of the given attributes.
<attributes>	mandatory	List of attributes used to compose the friendly name. Note that the order matters.
<attribute>	at least one	
<order>	opti	Defines the

	onal	default sort order for the class, if omitted the class is sorted on the friendly name
<columns>	mandatory	
<column id="name" ascending="true|false">	at least one	Either sort ascending or descending on this attribute. The order of the attributes is important.
<display_template>	optional	
<icon>	optional	Specify an icon for your class
<reconciliation>	mandatory	Define the default reconciliatio

		n scheme for data import.
<attributes>	mandatory	List of attributes used for the reconciliation.
<attribute>	at least one	
<fields>	mandatory	Declaration of attributes (cumulated with attributes inherited from a parent class, if any)
<field>	at least one	Declaration of an attribute. See the various types of attributes in section Attributes
<lifecycle>	opti	Lifecycle:

	onal	states and transitions
<attribute>name<attribute>	mandatory	Attribute used for the state (must be declared as AttributeEnum, can be inherited from a parent class)
<stimuli>	mandatory	List of events to which the object will be sensitive
<stimulus>	at least one	Declaration of a stimulus. See the various types of stimuli in section Stimuli
<states>	mandatory	Possible states. The list of states must be a subset of the values defined for the

		″state″ attribute.
<state id=″name″>	at least one	The complete definition of a state: name, attribute flags and transitions to other states
<initial_state_path>	optional	An ordered list of states through which the object will go when it is created
<state_ref>new<state_ref>	mandatory	A state
<flags>	mandatory	Define here how the attributes will be shown in the GUI, for the given state
<attribute id=″name″>	at	Combination of

	leas t one	flags interpreted by the GUI
<hidden>	opti onal	The attribute is hidden in this state
<read_only>	opti onal	The attribute is shown and cannot be modified
<must_prompt>	opti onal	The GUI must prompt the user when the object is entering the state
<must_change>	opti onal	The attribute must be changed by the user when the object is entering the state
<mandatory>	opti onal	The attribute must be defined in

		this state. This overrides the definition of the attribute (tag ″is_null_allowed″)
<transitions>	mandatory	List of possible transitions from this state to another
<transition>	at least one	For a given stimulus, defines the target state and the actions to perform
<stimulus>name<stimulus>	mandatory	Event triggering this transition
<target>name<target>	mandatory	State reached after the transition

<actions>	mandatory	Ordered list of actions to perform during the transition
<action>	at least one	A single action
<verb>name<verb>	mandatory	Name of the method that will be called (see the tag "methods" hereafter)
<methods>	mandatory	Additional function declarations. The function will be declared within the class. This is the mean to overload some functions of DBObject or cmdbAbstractObject. Use with care.
<method id="name">	zero	A function

	or more	
<comment>	optional	PHP comment. Will be predended to the declaration of the class into the generated (compiled) PHP code.
<static>false<static>	mandatory	Set to true if you need to declare a static function, false otherwise.
<access>public<access>	mandatory	Set to public, protected or private (See the documentation of PHP)
<type>Overload-DBObject<type>	mandatory	Use one of the following values: LifecycleAction,

		Overload-cmdbAbstractObject, Overload-iDisplay, Overload-DBObject, Overload-ExNihilo, Custom
<code>	mandatory	PHP code. Must include the function prototype. It is higly recommended to put it within a CDATA to avoid the need for escaping xml entities within your code.
<presentation>	mandatory	
<details>	mandatory	Defines the presentation for both the vizualisation and the edition form of an object.

			Can be overriden by the lifecycle flag ″hidden″
<items>		at least one	Refer to Presentation (details)
	<search>	mandatory	Ordered list of attributes shown in search forms
<items>		at least one	Refer to Presentation (search or list)
	<list>	mandatory	Ordered list of attributes shown by default in result lists
<items>		at least one	Refer to Presentation (search or list)

<menus>	mandatory	Declaration of the menus shown in the main GUI (left pane)
<menu>	at least one	Declaration of a menu. See the various types of menus in section Menus
<user_rights>	mandatory	Implementation of the user rights policy: users will have one or more profiles, granting them access rights.
<groups>	mandatory	Groups are sets of classes. Grants are given based on this grouping - see "profiles"
<group id="name">	at least one	A set of classes.

<classes>	mandatory	List of classes found in the group
<class id="name">	zero or more	
<profiles>	mandatory	Profiles that will be listed in the application. A user can have one or more profiles. The profile "administrator" is always present and cannot be redefined.
<profile id="123">	at least one	A usage profile.
<name>Configuration Manager<name>	mandatory	Name of the profile as it will be shown in the application. No translation is possible.

<description>Person in charge of the documentation of the managed CIs<description>	mandatory	Description of the profile (one line) as it will be shown in the application. No translation is possible.
<groups>	mandatory	Grants associated to the profile
<group>	at least one	Group for which grants must be defined
<actions>	mandatory	Allowed actions for the profile/group
<action>	at least one	See the various types of grant in section Action grants

Fields

The fields (also called attributes) are the actual data members of the objects. A field generally corresponds to one (or more) columns(s) in one table in the database. The different types of fields are listed in the table below:

Use the triangle arrow on the left of an item to expand/collapse its definition.

Tag	Usage	Description
<field id="name" xsi:type="AttributeString">	zero or more	A string, limited to one line of 255 characters
<sql>name<sql>	mandatory	The column used to store the value into the MySQL database
<default_value>name<default_value>	mandatory	The default value (can be specified as an empty string)
<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise
<field id="name" xsi:type="AttributeEnum">	zero or	An string that can take its value out

	more	of a fixed set of possible values
<values>	mandatory	List of possible values
<value>name<value>	at least one	Value. Must be made of alphanumeric characters. Other authorized characters: '_', '-'
<sql>name<sql>	mandatory	The column used to store the value into the MySQL database
<default_value>name<default_value>	mandatory	The default value (must be in the list of possible values)
<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise

<field id="name" xsi:type="AttributeEmailAddress">	zero or more	An email address
<sql>name<sql>	mandatory	The column used to store the value into the MySQL database
<default_value>name<default_value>	mandatory	The default value (can be specified as an empty string)
<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise
<field id="name" xsi:type="AttributeExternalKey">	zero or more	An external key: a pointer to an object of the given class
<sql>name<sql>	mandatory	The column used to store the value into the MySQL database

<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise
<on_target_delete>DEL_AUTO<on_target_delete>	mandatory	Define how the deletion of the target object will impact the current object. Allowed values are 'DEL_MANUAL' and 'DEL_AUTO'
<target_class>name<target_class>	mandatory	To class of the objects to which the external key is pointing
<filter>SELECT Location AS L WHERE L.org_id = :this->org_id<filter>	optional	OQL query to define a set of object to which the external key can point to. Use :this->*name* to refer to a value in the current object
<dependencies>	optional	Attributes on which the current attribute

		depends. This will be taken into account in the forms.
<attribute id="name">	mandatory	An attribute code
<max_combo_length>50<max_combo_length>	optional	The maximum number of elements in a drop-down list. If more then an autocomplete will be used. Defaults to the value given in the configuration file.
<min_autocomplete_chars>3<min_autocomplete_chars>	optional	The minimum number of characters to type in order to trigger the "autocomplete" behavior. Defaults to the value given in the configuration file.
<allow_target_creation>true<allow_target_creation>	optional	Displays the + button on external keys to create

		target objects. Defaults to the value given in the configuration file.
<field id="name" xsi:type="AttributeHierarchicalKey">	zero or more	An external key pointing to the same class, in order to build hierarchies of objects
<sql>name<sql>	mandatory	The column used to store the value into the MySQL database
<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise
<on_target_delete>DEL_AUTO<on_target_delete>	mandatory	Define how the deletion of the target object will impact the current object. Allowed values are 'DEL_MANUAL' and 'DEL_AUTO'

<filter>SELECT Location AS L WHERE L.org_id = :this->org_id<filter>	optio nal	OQL query to define a set of object to which the key can point to. Use :this->*name* to refer to a value in the current object
<dependencies>	optio nal	Attributes on which the current attribute depends. This will be taken into account in the forms.
<attribute id="name">	manda tory	An attribute code
<max_combo_length>50<max_combo _length>	optio nal	The maximum number of elements in a drop-down list. If more then an autocomplete will be used. Defaults to the value given in the configuration file.
<min_autocomplete_chars>3<min_	optio	The minimum number

autocomplete_chars>	nal	of characters to type in order to trigger the "autocomplete" behavior. Defaults to the value given in the configuration file.
<allow_target_creation>true<allow_target_creation>	optional	Displays the + button on external keys to create target objects. Defaults to the value given in the configuration file.
<field id="name" xsi:type="AttributeExternalField">	zero or more	An alias to an attribute hold by another object (see "ExternalKey")
<extkey_attcode>name<extkey_attcode>	mandatory	External key pointing to the remote class. This attribute must be defined in the current class.
<target_attcode>name<target_at	manda	The attribute of

tcode>	tory	the remote class
<field id="name" xsi:type="AttributeText">	zero or more	A multi-line text (limited to 64 Kb)
<sql>name<sql>	manda tory	The column used to store the value into the MySQL database
<default_value>name<default_va lue>	manda tory	The default value (can be specified as an empty string)
<is_null_allowed>true<is_null_ allowed>	manda tory	Set to "true" to let users leave this value undefined, false otherwise
<field id="name" xsi:type="AttributeLongText">	zero or more	A huge text (limited to 4 Gb)
<sql>name<sql>	manda tory	The column used to store the value into the MySQL

		database
<default_value>name<default_value>	mandatory	The default value (can be specified as an empty string)
<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise
<field id="name" xsi:type="AttributeLinkedSet">	zero or more	A set of objects pointing to the current object
<linked_class>name<linked_class>	mandatory	A class of objects having an external key pointing to the current object
<ext_key_to_me>name<ext_key_to_me>	mandatory	An external key attribute, defined on the linked class
<tracking_level>list<tracking_level>	optional	Adjust the recording of

		changes (history tab). Possibe values: none, list (track added and removed items), details (track modified items), all. Default to 'list'
<edit_mode>actions<edit_mode>	optional	Define the type of GUI for editing this link set. Possibe values: none, add_only, actions, in_place. Defaults to 'actions'
<count_min>0<count_min>	optional	unused yet
<count_max>0<count_max>	optional	unused yet
<field id="name" xsi:type="AttributeLinkedSetIndirect">	zero or more	A set of objects related to the current object by the mean of a "link class"

<linked_class>name<linked_class>	mandatory	A class of objects having an external key pointing to the current object
<ext_key_to_me>name<ext_key_to_me>	mandatory	An external key attribute, defined on the linked class
<ext_key_to_remote>name<ext_key_to_remote>	mandatory	An external key attribute, defined on the linked class, and pointing to the remote object
<tracking_level>list<tracking_level>	optional	Adjust the recording of changes (history tab). Possibe values: none, list (track added and removed items), details (track modified items), all. Default to 'all'
<duplicates>true<duplicates>	optional	Set to 'true' to allow duplicates. Defaults to 'false'

<count_min>0<count_min>	optio nal	unused yet
<count_max>0<count_max>	optio nal	unused yet
<field id="name" xsi:type="AttributeBlob">	zero or more	A blob, i.e. a binary string (limited to 4Gb). The name of the attribute is used as the prefix to name the columns used for storing the data.
<is_null_allowed>true<is_null_allowed>	optio nal	Set to "true" to let users leave this value undefined (default), false otherwise
<field id="name" xsi:type="AttributeInteger">	zero or more	An integer value
<sql>name<sql>	manda tory	The column used to store the value into the MySQL database

<default_value>name<default_value>	mandatory	The default value
<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise
<field id="name" xsi:type="AttributeDate">	zero or more	A date (no time)
<sql>name<sql>	mandatory	The column used to store the value into the MySQL database
<default_value>name<default_value>	mandatory	The default value (can be specified as an empty string)
<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise

<field id="name" xsi:type="AttributeDateTime">	zero or more	A date and time
<sql>name<sql>	manda tory	The column used to store the value into the MySQL database
<default_value>name<default_va lue>	manda tory	The default value (can be specified as an empty string)
<is_null_allowed>true<is_null_ allowed>	manda tory	Set to "true" to let users leave this value undefined, false otherwise
<field id="name" xsi:type="AttributeIPAddress">	zero or more	An IP address
<sql>name<sql>	manda tory	The column used to store the value into the MySQL database

<default_value>name<default_value>	mandatory	The default value (can be specified as an empty string)
<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise
<field id="name" xsi:type="AttributeURL">	zero or more	An URL (http...)
<sql>name<sql>	mandatory	The column used to store the value into the MySQL database
<default_value>name<default_value>	mandatory	The default value (can be specified as an empty string)
<is_null_allowed>true<is_null_allowed>	mandatory	Set to "true" to let users leave this value undefined, false otherwise

<target>_blank<target>	manda tory	Target attribute as it will be set into the A tag (see HTML specs)
<field id="name" xsi:type="AttributeStopWatch">	zero or more	Cumulate the time spent in some states
<states>	manda tory	States in which the stop-watch will be running
<state id="name">	zero or more	A state (as declared in the life-cycle of the class)
<goal>name<goal>	optio nal	Name of the class handling the computation of the time limit. Defaults to 'DefaultMetricComputer'
<working_time>name<working_time>	optio nal	Name of the class handling the computation of active times. Defaults to

		'DefaultWorkingTimeComputer'
<thresholds>	mandatory	Intermediate milestones, defined as a portion of the overall time goal.
<threshold>	zero or more	A milestone
<percent>80<percent>	mandatory	Position of the milestone, relative to the overall duration limit
<actions>	mandatory	What must be done when the milestone is being passed.
<action>	zero or more	A milestone
<verb>DoThis<verb>	mandatory	Function (of the current PHP class)

<params>	optio nal	Arguments to be passed to the function
<param>	zero or more	A scalar argument (number of string)

Presentation (details)

The presentation “details” defines the structure of the form used to enter an object and to display its “details”. It can be a simple list (in which case the fields are displayed in one column), but can also define columns and fieldsset to group related fields together.

Use the triangle arrow on the left of an item to expand/collapse its definition.

Tag	Usage	Description
<items>	mandatory	
<item>	at least one	An item can be either: an attribute (id = attribute code), a column (id = col:number) or a field set (id = fieldset:dictionary entry)
<rank>123<rank>	mandatory	Display rank. Item are

		ordered from top to bottom, left to right. This must be an integer value.
<items>	optional	In case the item defined above is a column or a field set, then this tag must be defined to contain the items (recursively, though this recursion is limited).

Presentation (search or list)

A simple ordered list of fields used when displaying lists of object (usage = list) or for displaying the search form for a given class (usage = seach)

Use the triangle arrow on the left of an item to expand/collapse its definition.

Tag	Usage	Description
<items>	mandatory	An ordered list of attributes
<item id="name">	at least one	An attribute
<rank>123<rank>	mandatory	Rank of the attribute in the list (integer)

Stimuli

Each transition from one state to another (for objects with a life -cycle) is trigerred when the object receives a stimuli. The different types of stimuli are listed in the table below:

Tag	Usage	Description
<stimulus id="name" xsi:type="StimulusUserAction">	at least one	An action decided by the end-user. The stimuli is displayed in the Actions menu (if the object is in a state for which this stimuli is taken into account
<stimulus id="name" xsi:type="StimulusInternal">	at least one	An action that can be triggered programmatically.

Menus

The different types of menus are listed in the table below:

Use the triangle arrow on the left of an item to expand/collapse its definition.

Tag	Usage	Description

<menu id="name" xsi:type="MenuGroup">	optional	Top level menu. This menu will always remain visible. It is a container for other menus.
<rank>123.45<rank>	mandatory	Display rank. This is a float. Menus are ordered by ascending rank: the smallest rank is on top.
<enable_admin_only>1<enable_admin_only>	optional	If set to '1' then only administrators will see this menu entry
<menu id="name" xsi:type="DashboardMenuNode">	optional	Dashboard. The contents can be produced by the mean of the "export" function. The contents of the dashboard can be found in a file (definition_file) or directly here (definition)

<rank>123.45<rank>	mandatory	Display rank. This is a float. Menus are ordered by ascending rank: the smallest rank is on top.
<enable_admin_only>1<enable_admin_only>	optional	If set to'1' then only administrators will see this menu entry
<parent>name<parent>	mandatory	Parent menu node, either a top node or an intermediate node
<definition_file>name<definition_file>	optional	Dashboard definition file. The path is relative to the module in which the menu is declared.
<definition>name<definition>	optional	Dashboard definition contents. This tag will be ignored if the

		tag definition_file has been given and is not empty.
<menu id="name" xsi:type="NewObjectMenuNode">	optional	Shortcut to create a new object.
<rank>123.45<rank>	mandatory	Display rank. This is a float. Menus are ordered by ascending rank: the smallest rank is on top.
<enable_admin_only>1<enable_admin_only>	optional	If set to '1' then only administrators will see this menu entry
<parent>name<parent>	mandatory	Parent menu node, either a top node or an intermediate node
<class>name<class>	mandatory	Class of the object to create

<menu id="name" xsi:type="SearchMenuNode">	optional	Shortcut to search for objects.
<rank>123.45<rank>	mandatory	Display rank. This is a float. Menus are ordered by ascending rank: the smallest rank is on top.
<enable_admin_only>1<enable_admin_only>	optional	If set to '1' then only administrators will see this menu entry
<parent>name<parent>	mandatory	Parent menu node, either a top node or an intermediate node
<class>name<class>	mandatory	Class of the objects to search for.
<menu id="name" xsi:type="TemplateMenuNode">	optional	Deprecated. Kept for backward compatibility.

<rank>123.45<rank>	mandatory	Display rank. This is a float. Menus are ordered by ascending rank: the smallest rank is on top.
<enable_admin_only>1<enable_admin_only>	optional	If set to'1' then only administrators will see this menu entry
<parent>name<parent>	mandatory	Parent menu node, either a top node or an intermediate node
<template_file>name<template_file>	optional	Dashboard definition file. The path is relative to the module in which the menu is declared.
<menu id="name" xsi:type="OQLMenuNode">	optional	Shortcut to display a search result, given an OQL query.

<rank>123.45<rank>	mandatory	Display rank. This is a float. Menus are ordered by ascending rank: the smallest rank is on top.
<auto_reload>standard<auto_reload>	optional	Determines how the display will be refreshed: "none" to disable this feature (default), "standard" or "fast" to refresh periodically based on the corresponding configuration setting, or "123" to refresh every 123 seconds.
<enable_admin_only>1<enable_admin_only>	optional	If set to '1' then only administrators will see this menu entry
<parent>name<parent>	mandatory	Parent menu node, either a top node or an intermediate node

<oql>SELECT UserRequest WHERE agent_id = :current_contact_id AND status NOT IN ("closed","resolved")<oql>	mandatory	Object query. The only context parameter available is "current_contact_id".
<do_search>1<do_search>	optional	If set to 1, then the search is executed by default when the user clicks on the menu.
<menu id="name" xsi:type="WebPageMenuNode">	optional	An hyperlink to a page internal or external to iTop.
<rank>123.45<rank>	mandatory	Display rank. This is a float. Menus are ordered by ascending rank: the smallest rank is on top.
<enable_admin_only>1<enable_admin_only>	optional	If set to '1' then only administrators will see this menu entry

<parent>name<parent>	mandatory	Parent menu node, either a top node or an intermediate node
<url>$$www.openitop.com/<url>	optional	URL to the page within the module. Prefix by a $$ to specify an absolute URL. Prefix by a $ to specify an URL relative to the iTop root URL. No prefix: relative to the module directory (buggy FIXME)
<menu id="name" xsi:type="ShortcutContainerMenuNode">	optional	Container for shortcuts. Only one container must be defined. The effects are unpredictable is several menus of this type are defined!
<rank>123.45<rank>	mandatory	Display rank. This is a float. Menus are ordered by ascending rank: the

		smallest rank is on top.
<enable_admin_only>1<enable_admin_only>	optional	If set to'1' then only administrators will see this menu entry
<parent>name<parent>	mandatory	Parent menu node, either a top node or an intermediate node

Action grants

Grants are the basic elements that define the rights associated with a specific action for a given profile. The different actions that can be "granted" are listed in the table below:

Tag	Usage	Description
<action xsi:type="read">allow<action>	optional	Read: set to "allow" or "deny"
<action xsi:type="write">allow<action>	optional	Write: set to "allow" or "deny"

<action xsi:type="delete">allow<action>	optional	Delete: set to "allow" or "deny"
<action xsi:type="bulk read">allow<action>	optional	Export data: set to "allow" or "deny"
<action xsi:type="bulk write">allow<action>	optional	Perform massive changes: set to "allow" or "deny"
<action xsi:type="bulk delete">allow<action>	optional	Perform bulk deletion: set to "allow" or "deny"

Extensions API

In addition to modifying the XML data model, it is possible to implement specific behaviors in iTop by the mean of so-called **extensions**.

Overview

An extension is made of PHP code. Your code will be invoked by iTop when building the answer to HTTP requests. For instance, it is possible to hook the display of an object to show more information.

In practice, your code consists in declaring a PHP class implementing an interface known by iTop as being an extension interface. iTop detects automatically the existence of your class and invokes the methods at some specific moments during its execution.

To make this happen, the code of your class must be in a file included by iTop. The recommended way for doing so is to put your code into a main.my-module.php file (see Content of a module).

An example

Let's imagine that we have web based application which provides some advanced reporting on the availability of Servers. We would like to provide an hyperlink so that end-users can quickly jump from the details of a Server in iTop into the corresponding report in the monitoring application.

One possible solution is to show this an hyperlink to the monitoring application into the "Actions" popup-menu on all Servers.

The implementation consists in implementing the interface iPopupMenuExtension:

main.mymodule.php

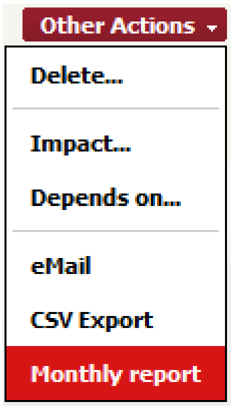
```
class MyPopupExtension implements iPopupMenuExtension
{
   public static function EnumItems($iMenuId, $param)
   {
      if ($iMenuId == self::MENU_OBJDETAILS_ACTIONS)
      {
         $oObject = $param;
         if ($oObject instanceof Server)
         {
             $sUID = 'MyPopupExtension-Monitoring'; // Make sure
that each menu item has a unique "ID"
             $sLabel = 'Monthly report';
             $sURL =
'http://myapp/show_report?server_fqdn='.$oObject->Get('name');
             $sTarget = '_blank';
             $oMenuItem = new URLPopupMenuItem($sUID, $sLabel,
$sURL, $sTarget);
             return array($oMenuItem);
         }
      }
      return array();
   }
}
```

The method EnumItems will be called by iTop in several circumstances. When displaying the details of an object, $params is the target object.

As our method will be called for any kind of object, we have to filter out the classes of objects that are not relevant for this action.

As you can see, one can handle several types of objects and several types of menus with the same extension (depending on $iMenuId and $param).

The outcome of this plugin is an additional menu entry on the details page of any server:



Interfaces reference documentation

API Reference for extensions

There are several interfaces for extending iTop. Each interface corresponds to a specific type of extension, as shown on the table below:

Interface	Description
iApplicationUIExtension	Implement this interface to change the behavior of the GUI for some objects (when displaying the details or editing an object).
iApplicationObjectExtension	Implement this interface to perform specific actions when objects are created, updated or deleted
iPageUIExtension	Implement this interface to add content to any iTopWebPage (web pages containing the iTop menu on the left)
iPopupMenuExtension	Add menu items in the "popup" menus inside iTop.

Adding a new field to the Server class

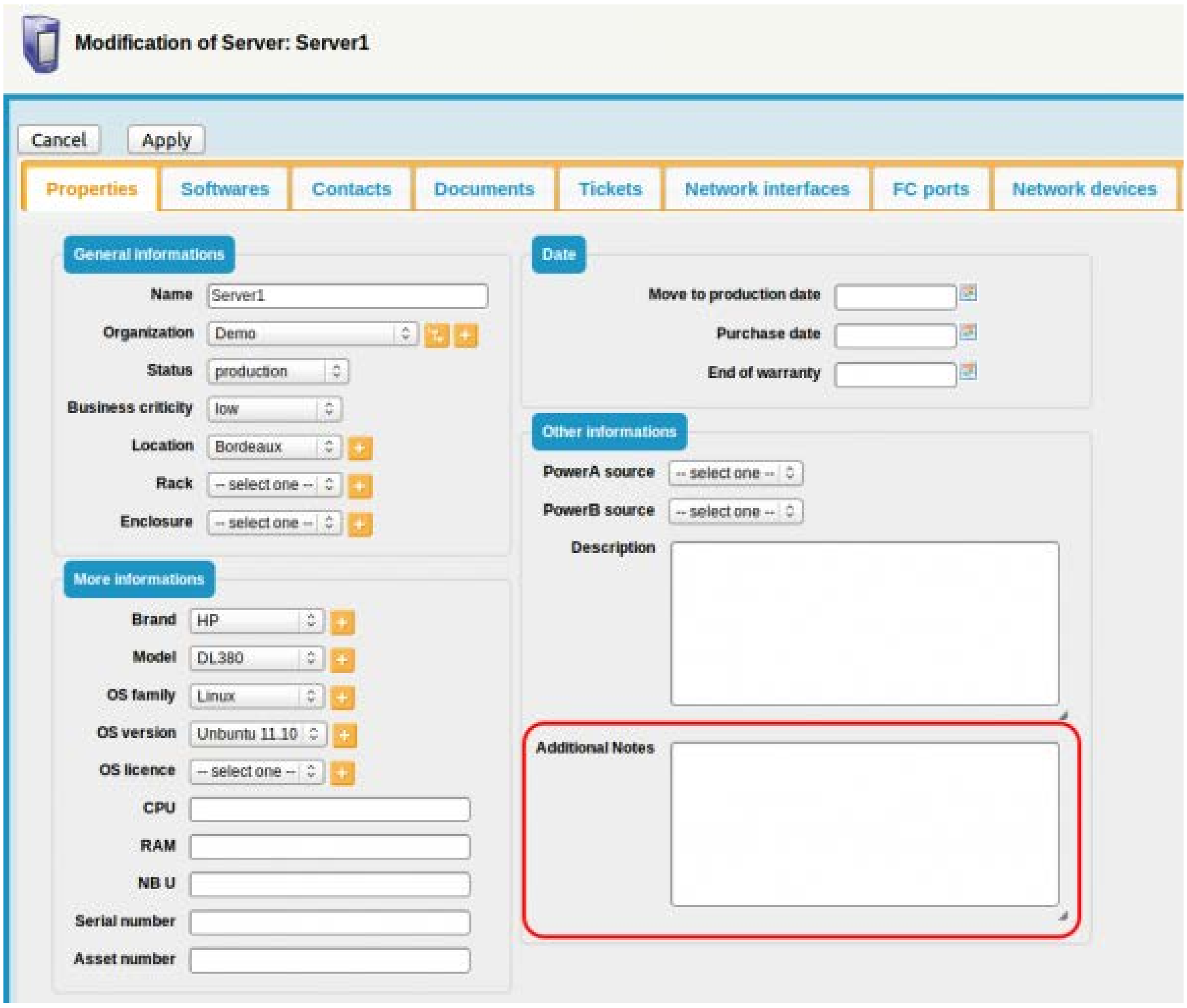
This document explains, step by step, how to create your own iTop module in order to add a new field to an existing iTop object.

Goals of this tutorial

In this step-by-step tutorial you will learn to:

- create your own extension module for iTop 2.0
- add a new field to an existing class of object

For the purpose of this tutorial we will add a text field labeled *Additional Notes* to the Server object.



Name	Server1
Organization	Demo
Status	production
Business criticity	low
Location	Bordeaux
Rack	-- select one --
Enclosure	-- select one --
Brand	HP
Model	DL380
OS family	Linux
OS version	Unbuntu 11.10
OS licence	-- select one --
CPU	
RAM	
NB U	
Serial number	
Asset number	
Move to production date	
Purchase date	
End of warranty	
PowerA source	-- select one --
PowerB source	-- select one --
Description	
Additional Notes	

What you will need

- iTop installed on a development machine, on which you can easily access/edit the files.

- A text editor capable of editing PHP and XML file and supporting UTF-8. On Windows you can use Wordpad (Notepad does not like Unix line endings) or one of the excellent free development IDEs like PSPad or Notepad++.

Customization process

The customization process is the following:

1. Install a development instance of iTop. It is always better not to experiment in production !!
2. Install the toolkit to assist you in the customization
3. Create a new (empty) module using the module creation wizard
4. Copy this new module in the `extensions` folder on iTop and run the setup again to install the empty module
5. Modify the module in `extensions` and use the toolkit to check your customizations

Repeat the last point until you are satisfied with your customization. When you are done, your new module is ready to be deployed. Copy the module folder in the extension directory on your production iTop instance and run the setup to install it.

Step by step tutorial

Create your customization module

Use the module creation wizard. Fill the form with the following values:

Label	Value	Remarks
Module name	sample-add-attribute	Names starting with `itop-` and `combodo-` are reserved for use by Combodo. It is recommended not to put spaces or accentuated characters in the name of the module. Two modules with the same name cannot co-exist in the same iTop instance.
Module Label	Add Attribute Sample	This label will be displayed in the setup wizard. Localized characters and spaces are allowed
Module Version	1.0.0	The convention is to use a 3 digits numbering scheme: X.Y.Z
Category	business	Modules that provide modifications to the data model should be in the category 'business'
Dependencies	itop-config-mgmt/2.0.0	Our customization module depends on the module iTop Configuration Management version 2.0.0 in which the

Server class is defined

Click **Generate !** to download the empty module as a zip file.

When a module modifies an existing class, it **must** be loaded after the module that declared the class to be modified. To achieve this, make sure that the first module is listed in the *dependencies* of your new module.

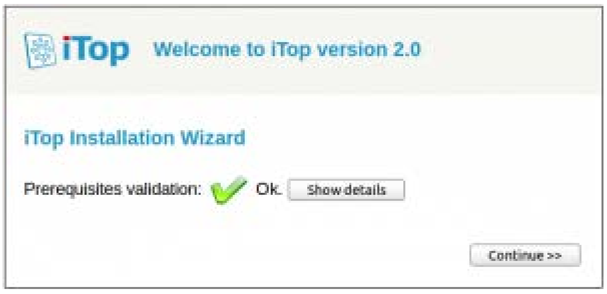
For example if you want to alter the definition of the VirtualMachine class, your custom module must depend on itop-virtualization-mgmt/2.0.0

Install the empty module

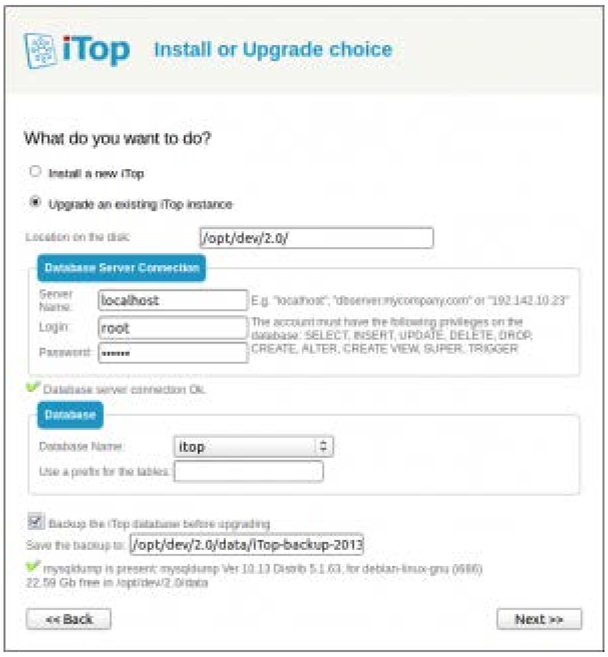
Expand the content of the zip into the extensions folder of your development iTop instance. You should now have a folder named sample-add-attribute inside the extensions folder. this folder contains the following files:

- datamodel.sample-add-attribute.xml
- module.sample-add-attribute.php
- en.dict.sample-add-attribute.php
- model.sample-add-attribute.php

Make sure that the file conf/production/config-itop.php is writable for the web server (on Windows: right click to display the file properties and uncheck the read-only flag; on Linux change the rights of the file), then launch the iTop installation by pointing your browser to **http://your_itop/setup/**



Click “Continue »” to start the re-installation.


iTop Install or Upgrade choice

What do you want to do?

- ○ Install a new iTop
- ● Upgrade an existing iTop instance

Location on the disk: /opt/dev/2.0/

Database Server Connection

Server Name: localhost E.g. "localhost", "dbserver.mycompany.com" or "192.142.10.23"

Login: root The account must have the following privileges on the database: SELECT, INSERT, UPDATE, DELETE, DROP, CREATE, ALTER, CREATE VIEW, SUPER, TRIGGER

Password: ••••••

✓ Database server connection Ok.

Database

Database Name: itop

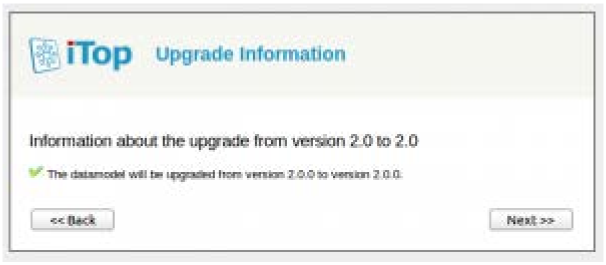
Use a prefix for the tables:

☑ Backup the iTop database before upgrading

Save the backup to: /opt/dev/2.0/data/iTop-backup-2013

✓ mysqldump is present: mysqldump Ver 10.13 Distrib 5.1.63, for debian-linux-gnu (i686)
22.59 Gb free in /opt/dev/2.0/data

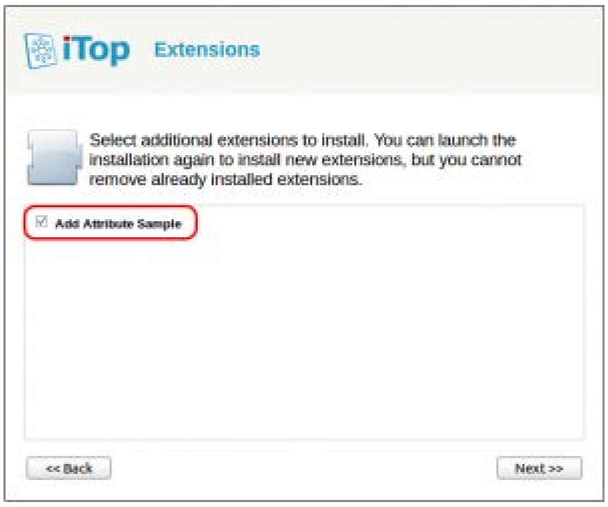
Make sure that “Update an existing instance” is selected before clicking “Next »”.


iTop Upgrade Information

Information about the upgrade from version 2.0 to 2.0

✓ The datamodel will be upgraded from version 2.0.0 to version 2.0.0.

Continue to the next steps of the wizard…


iTop Extensions

Select additional extensions to install. You can launch the installation again to install new extensions, but you cannot remove already installed extensions.

☑ Add Attribute Sample

Your custom module should appear in the list of “Extensions”. If it’s not the case, check that the module files were copied in the proper location and that the web server has enough rights to read them.

Select your custom module before clicking “Next »” and complete the installation.

Add a new field to the Server class

Using you favorite text editor, open the file datamodel.sample-add-attribute.xml.

Remove the tags <menus></menus> since the module will not contain any menu definition.

Inside the classes tag, add the following piece of code:

```
    <class id="Server">
      <fields>
        <field id="notes" xsi:type="AttributeText" _delta="define">
          <sql>notes</sql>
          <default_value/>
          <is_null_allowed>true</is_null_allowed>
        </field>
      </fields>
     </class>
```

This instructs iTop to modify the existing class “Server” by adding a new field (notice the _delta=“define” on the field tag) of type AttributeText. This new field is named notes (since it is defined with id=“notes”). The corresponding values will be stored in the database in the column notes (thanks to the definition <sql>notes</sql>).

For more information about the meaning of the various parameters of the field tag (and also for the list of all possible types of fields) refer to the XML reference documentation.

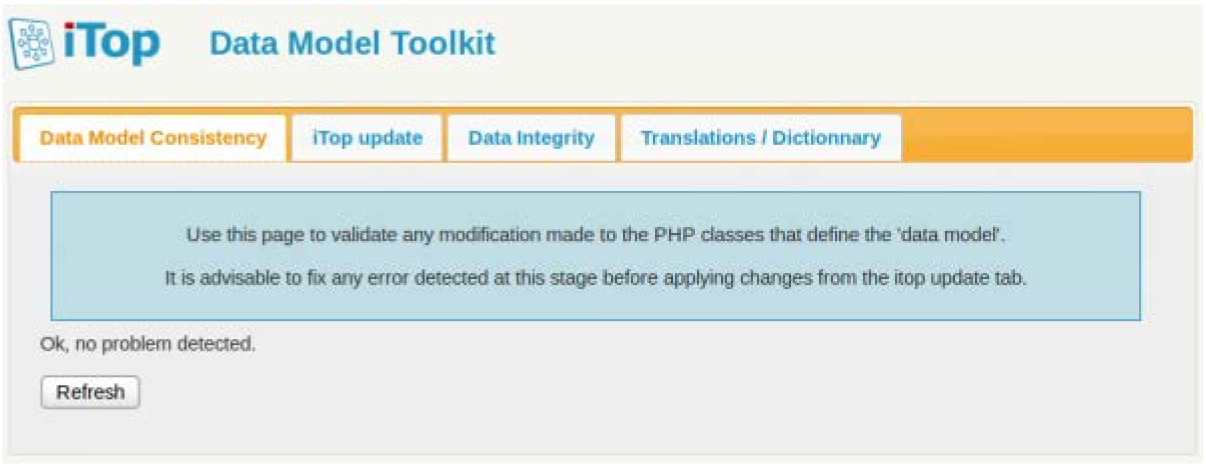
You should now have the following XML file:

datamodel.sample-add-attribute.xml

```
      <?xml version="1.0" encoding="UTF-8"?>
      <itop_design
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      version="1.0">
        <classes>
```

```
    <class id="Server">
      <fields>
        <field id="notes" xsi:type="AttributeText"
_delta="define">
          <sql>notes</sql>
          <default_value/>
          <is_null_allowed>true</is_null_allowed>
        </field>
      </fields>
     </class>
   </classes>
</itop_design>
```

Check your modification by running the toolkit. Point your browser to **http://your_itop/toolkit**.



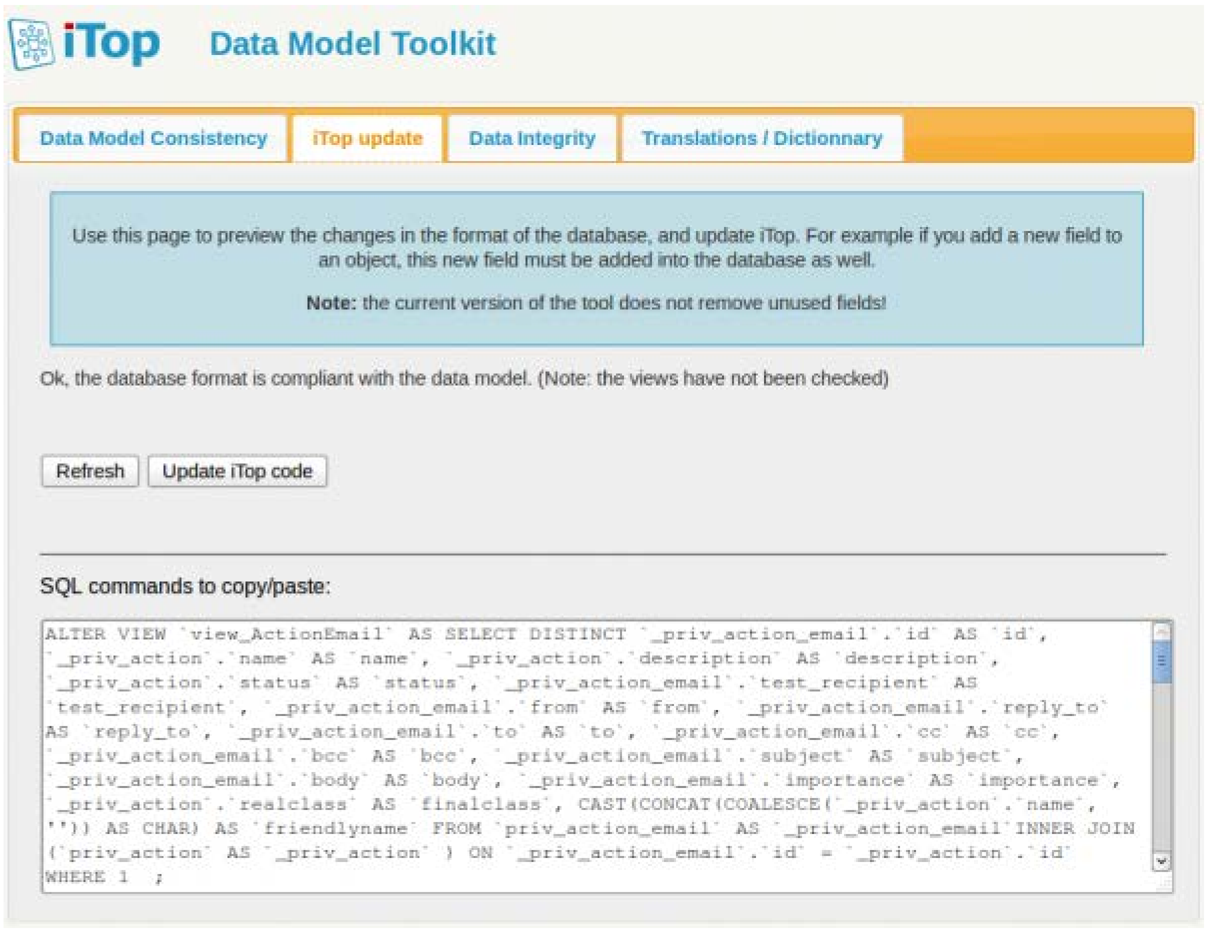
Use this page to validate any modification made to the PHP classes that define the 'data model'.

It is advisable to fix any error detected at this stage before applying changes from the itop update tab.

Ok, no problem detected.

If any error is reported at this stage, fix them by editing the XML file and check again your modifications by clicking on the “Refresh” button in the toolkit page.

Once all the errors have been fixed, you can apply the modifications to iTop by using the second tab of the toolkit:



Use this page to preview the changes in the format of the database, and update iTop. For example if you add a new field to an object, this new field must be added into the database as well.

Note: the current version of the tool does not remove unused fields!

Ok, the database format is compliant with the data model. (Note: the views have not been checked)

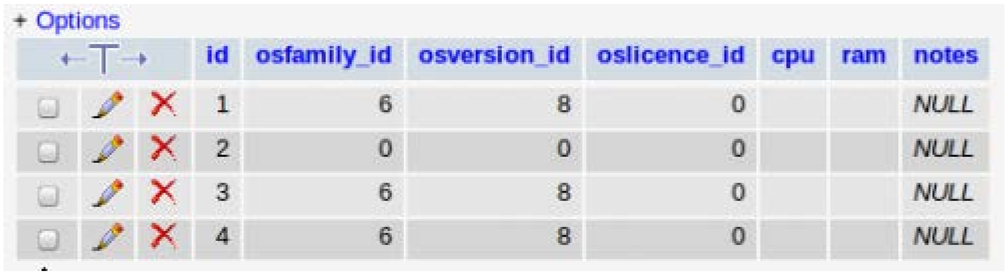
SQL commands to copy/paste:

```
ALTER VIEW `view_ActionEmail` AS SELECT DISTINCT `_priv_action_email`.`id` AS `id`,
`_priv_action`.`name` AS `name`, `_priv_action`.`description` AS `description`,
`_priv_action`.`status` AS `status`, `_priv_action_email`.`test_recipient` AS
`test_recipient`, `_priv_action_email`.`from` AS `from`, `_priv_action_email`.`reply_to`
AS `reply_to`, `_priv_action_email`.`to` AS `to`, `_priv_action_email`.`cc` AS `cc`,
`_priv_action_email`.`bcc` AS `bcc`, `_priv_action_email`.`subject` AS `subject`,
`_priv_action_email`.`body` AS `body`, `_priv_action_email`.`importance` AS `importance`,
`_priv_action`.`realclass` AS `finalclass`, CAST(CONCAT(COALESCE(`_priv_action`.`name`,
'')) AS CHAR) AS `friendlyname` FROM `priv_action_email` AS `_priv_action_email`INNER JOIN
(`priv_action` AS `_priv_action` ) ON `_priv_action_email`.`id` = `_priv_action`.`id`
WHERE 1  ;
```

Click on the button **Update iTop Code** to:

1. Compile the XML data model to PHP classes
2. Update the database schema to add the new `text` column.

At this point, if you look at the schema of the MySQL database, you can see the additional “notes” column added to the “server” table. However if you navigate to a Server in iTop, nothing has changed.



	id	osfamily_id	osversion_id	oslicence_id	cpu	ram	notes
	1	6	8	0			NULL
	2	0	0	0			NULL
	3	6	8	0			NULL
	4	6	8	0			NULL

This is because iTop was not instructed how to display the added field. So the field exists but is not displayed in iTop.

Make the new field visible

Let's add the new field to the "details" of the Server object, just below the "Description". This can be achieved by redefining the way the "details" of a Server are displayed.

Using your text editor, open the file datamodels/2.x/itop-config-mgmt/datamodel.itop-config-mgmt.xml.

Search for the string **<class id="Server"** to locate the definition of the Server class.

Scroll down to the <presentation> tag and copy the whole content of the <details>…</details> tag.

Paste this whole definition in datamodel.sample-add-attribute.xml after the closing </field> tag, and enclose it in <presentation>…</presentation> tags.

Change the opening tag <details> to <details _delta="redefine"> in order to instruct iTop to *redefine* the presentation for the "details".

Insert the 3 lines:

```
                    <item id="notes">
                      <rank>40</rank>
                    </item>
```

Just after the lines:

```
                    <item id="description">
                      <rank>30</rank>
                    </item>
```

You should now obtain the following XML file:

datamodel.sample-add-attribute.xml

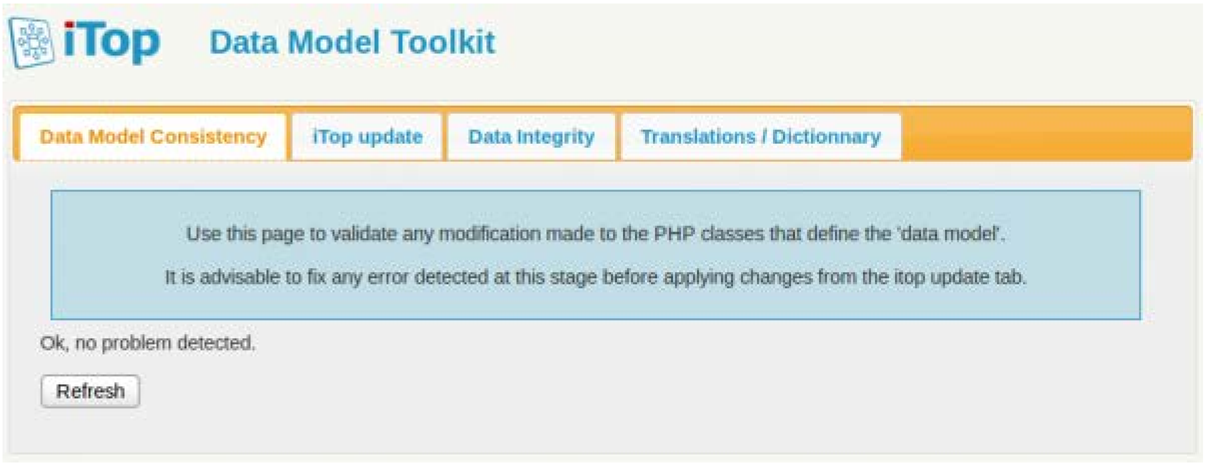
```
      <?xml version="1.0" encoding="UTF-8"?>
      <itop_design
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      version="1.0">
        <classes>
          <class id="Server">
            <fields>
              <field id="notes" xsi:type="AttributeText"
      _delta="define">
                <sql>notes</sql>
                <default_value/>
                <is_null_allowed>true</is_null_allowed>
              </field>
            </fields>
            <presentation>
              <details _delta="redefine">
                <items>
                  <item id="softwares_list">
                    <rank>10</rank>
                  </item>
                  <item id="contacts_list">
                    <rank>20</rank>
                  </item>
                  <item id="documents_list">
                    <rank>30</rank>
                  </item>
                  <item id="tickets_list">
                    <rank>40</rank>
                  </item>
                  <item id="physicalinterface_list">
                    <rank>50</rank>
                  </item>
                  <item id="fiberinterfacelist_list">
                    <rank>60</rank>
                  </item>
                  <item id="networkdevice_list">
                    <rank>70</rank>
                  </item>
                  <item id="san_list">
                    <rank>80</rank>
                  </item>
                  <item id="logicalvolumes_list">
                    <rank>90</rank>
                  </item>
                  <item id="providercontracts_list">
                    <rank>100</rank>
                  </item>
                  <item id="services_list">
                    <rank>110</rank>
                  </item>
                  <item id="col:col1">
```

```
<rank>120</rank>
<items>
  <item id="fieldset:Server:baseinfo">
    <rank>10</rank>
    <items>
      <item id="name">
        <rank>10</rank>
      </item>
      <item id="org_id">
        <rank>20</rank>
      </item>
      <item id="status">
        <rank>30</rank>
      </item>
      <item id="business_criticity">
        <rank>40</rank>
      </item>
      <item id="location_id">
        <rank>50</rank>
      </item>
      <item id="rack_id">
        <rank>60</rank>
      </item>
      <item id="enclosure_id">
        <rank>70</rank>
      </item>
    </items>
  </item>
  <item id="fieldset:Server:moreinfo">
    <rank>20</rank>
    <items>
      <item id="brand_id">
        <rank>10</rank>
      </item>
      <item id="model_id">
        <rank>20</rank>
      </item>
      <item id="osfamily_id">
        <rank>30</rank>
      </item>
      <item id="osversion_id">
        <rank>40</rank>
      </item>
      <item id="oslicence_id">
```

```
          <rank>50</rank>
        </item>
        <item id="cpu">
          <rank>60</rank>
        </item>
        <item id="ram">
          <rank>70</rank>
        </item>
        <item id="nb_u">
          <rank>80</rank>
        </item>
        <item id="serialnumber">
          <rank>90</rank>
        </item>
        <item id="asset_number">
          <rank>100</rank>
        </item>
      </items>
    </item>
  </items>
</item>
<item id="col:col2">
  <rank>130</rank>
  <items>
    <item id="fieldset:Server:Date">
      <rank>10</rank>
      <items>
        <item id="move2production">
          <rank>10</rank>
        </item>
        <item id="purchase_date">
          <rank>20</rank>
        </item>
        <item id="end_of_warranty">
          <rank>30</rank>
        </item>
      </items>
    </item>
    <item id="fieldset:Server:otherinfo">
      <rank>20</rank>
      <items>
        <item id="powerA_id">
          <rank>10</rank>
        </item>
```

```
                        <item id="powerB_id">
                          <rank>20</rank>
                        </item>
                        <item id="description">
                          <rank>30</rank>
                        </item>
                        <item id="notes">
                          <rank>40</rank>
                        </item>
                      </items>
                    </item>
                  </items>
                </item>
              </items>
            </details>
          </presentation>
        </class>
      </classes>
    </itop_design>
```

Check your modification by running the toolkit. Point your browser to **http://your_itop/toolkit**.



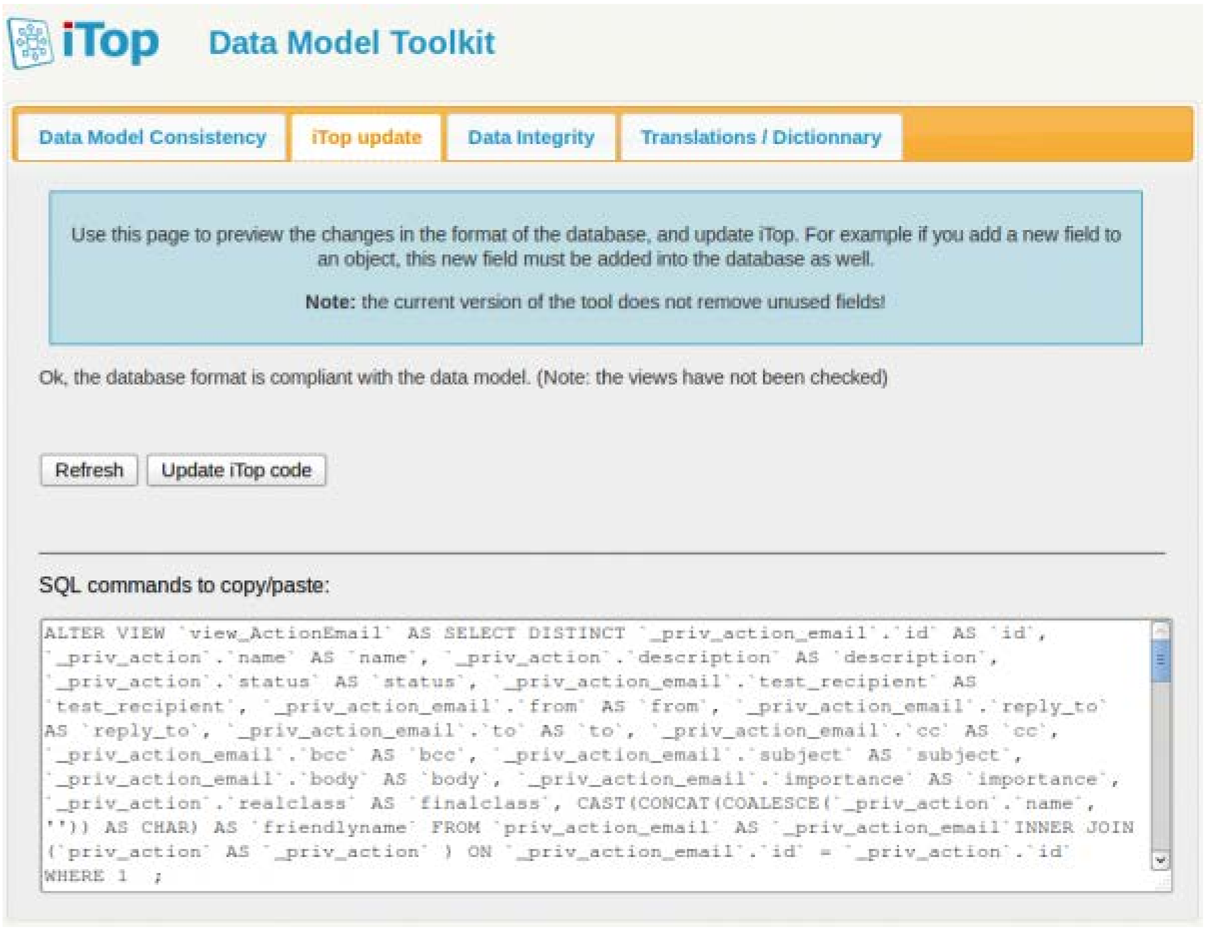
Use this page to validate any modification made to the PHP classes that define the 'data model'.

It is advisable to fix any error detected at this stage before applying changes from the itop update tab.

Ok, no problem detected.

If any error is reported at this stage, fix it by editing the XML file and check again your modifications by clicking on the “Refresh” button in the toolkit page.

Once all the errors have been fixed, you can apply the modifications to iTop by using the second tab of the toolkit:



Use this page to preview the changes in the format of the database, and update iTop. For example if you add a new field to an object, this new field must be added into the database as well.

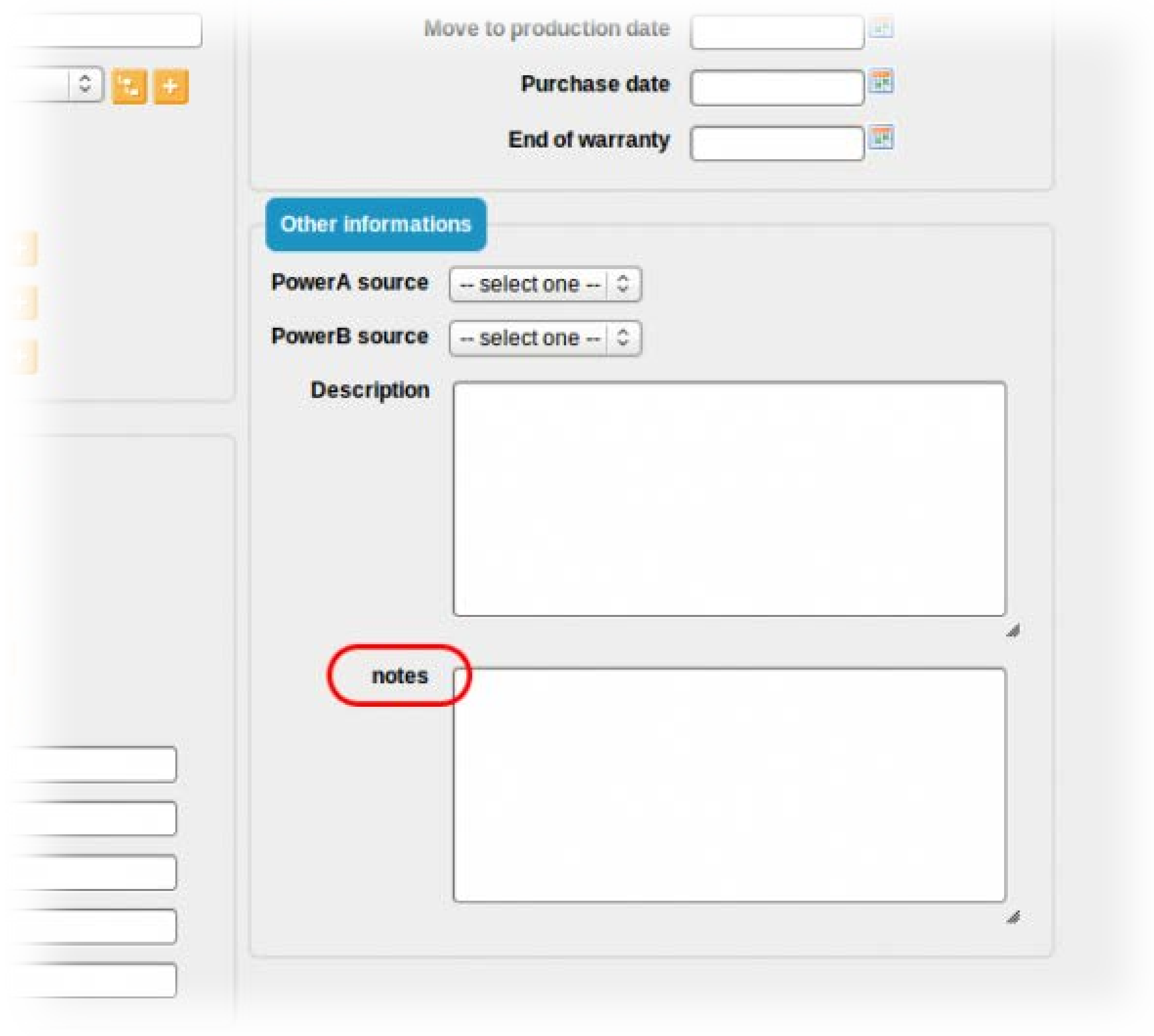
Note: the current version of the tool does not remove unused fields!

Ok, the database format is compliant with the data model. (Note: the views have not been checked)

SQL commands to copy/paste:

```
ALTER VIEW `view_ActionEmail` AS SELECT DISTINCT `_priv_action_email`.`id` AS `id`,
`_priv_action`.`name` AS `name`, `_priv_action`.`description` AS `description`,
`_priv_action`.`status` AS `status`, `_priv_action_email`.`test_recipient` AS
`test_recipient`, `_priv_action_email`.`from` AS `from`, `_priv_action_email`.`reply_to`
AS `reply_to`, `_priv_action_email`.`to` AS `to`, `_priv_action_email`.`cc` AS `cc`,
`_priv_action_email`.`bcc` AS `bcc`, `_priv_action_email`.`subject` AS `subject`,
`_priv_action_email`.`body` AS `body`, `_priv_action_email`.`importance` AS `importance`,
`_priv_action`.`realclass` AS `finalclass`, CAST(CONCAT(COALESCE(`_priv_action`.`name`,
'')) AS CHAR) AS `friendlyname` FROM `priv_action_email` AS `_priv_action_email`INNER JOIN
(`priv_action` AS `_priv_action` ) ON `_priv_action_email`.`id` = `_priv_action`.`id`
WHERE 1  ;
```

If you now navigate to the details of a Server in iTop you should see the following:



Add a label for the new field

Notice that the label of the new field is iTop is notes (by default it is equal to the name of the field). In order to change this to Additional Notes we have to add an entry in the *dictionary*.

Using you text editor, open the file en.dict.sample-add-attribute.php.

Insert the line:

```
'Class:Server/Attribute:notes' => 'Additional Notes',
```

Just below the comment:

```
        // Dictionary entries go here
```

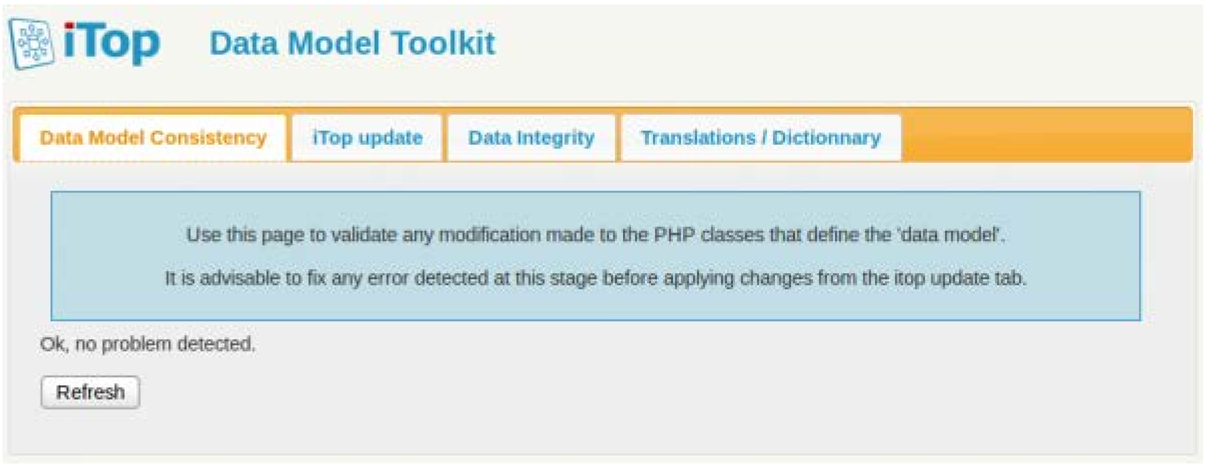
You should obtain the following file:

en.dict.sample-add-attribute.php

```
<?php
/**
 * Localized data
 *
 * @copyright   Copyright (C) 2013 Your Company
 * @license     http://opensource.org/licenses/AGPL-3.0
 */

Dict::Add('EN US', 'English', 'English', array(
  // Dictionary entries go here
  'Class:Server/Attribute:notes' => 'Additional Notes',
));
?>
```

One more time, check your modification by running the toolkit.



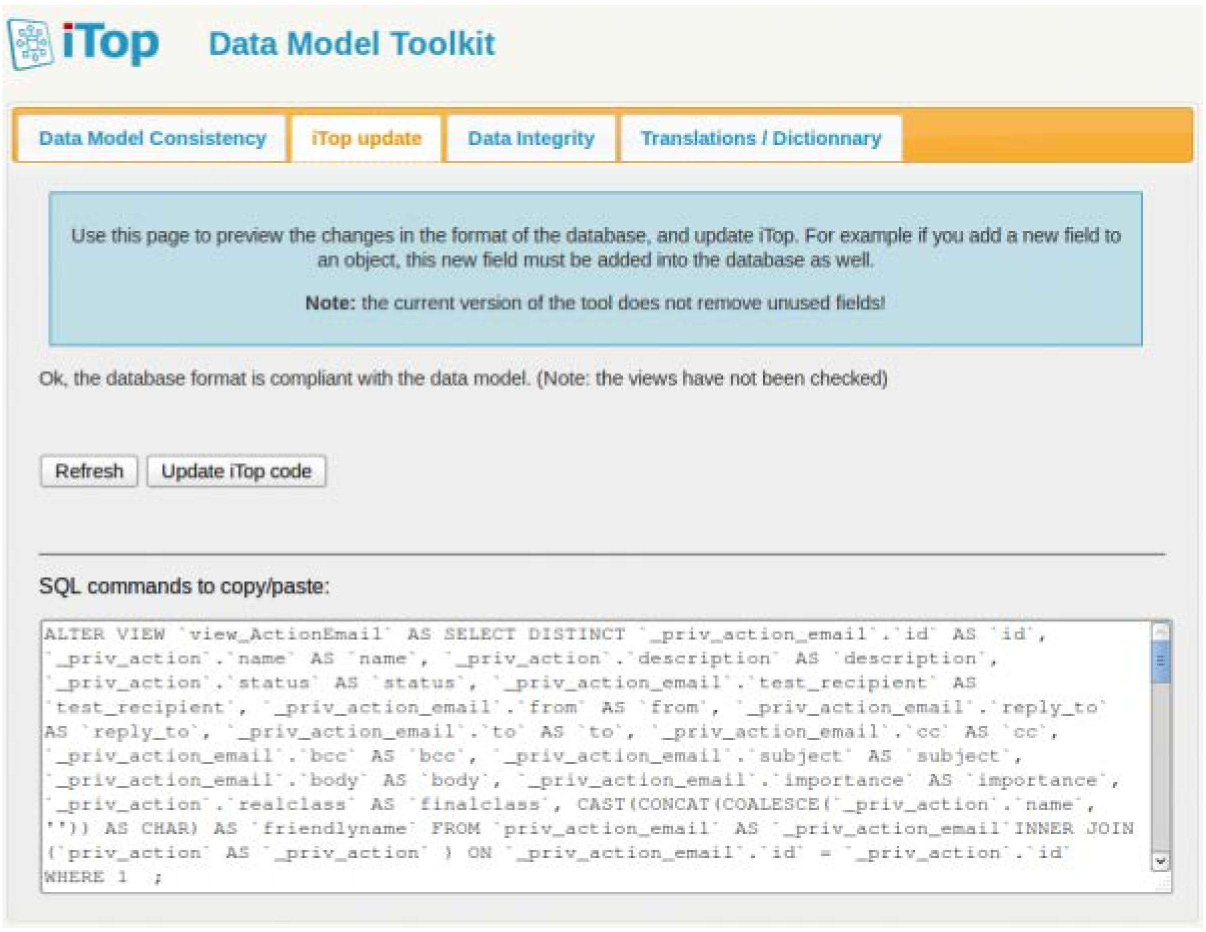
Use this page to validate any modification made to the PHP classes that define the 'data model'.

It is advisable to fix any error detected at this stage before applying changes from the itop update tab.

Ok, no problem detected.

If errors are reported at this stage, fix them by editing the PHP file and check again your modifications by clicking on the "Refresh" button in the toolkit page.

Once all the errors have been fixed, you can apply the modifications to iTop by using the second tab of the toolkit:



Use this page to preview the changes in the format of the database, and update iTop. For example if you add a new field to an object, this new field must be added into the database as well.

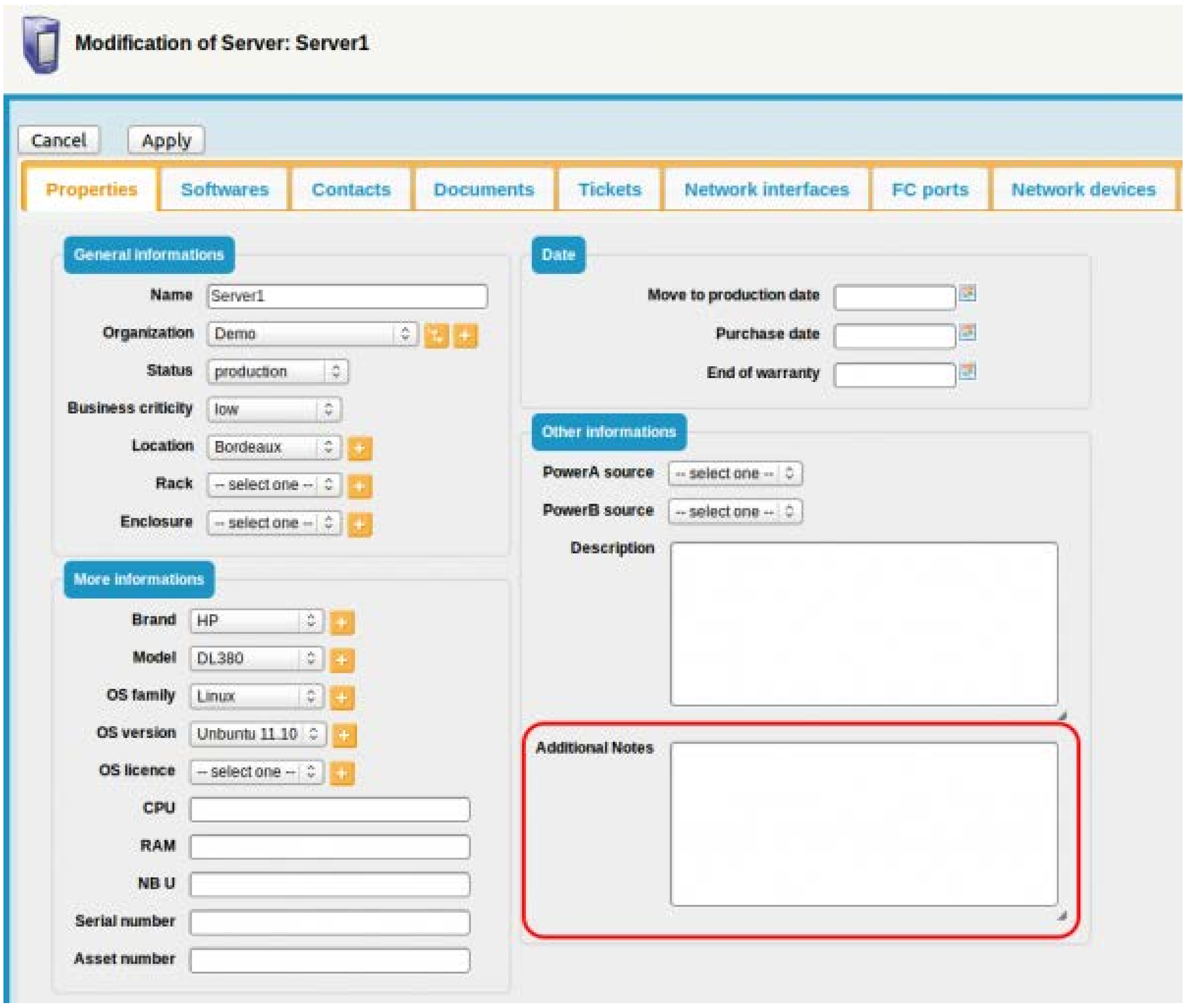
Note: the current version of the tool does not remove unused fields!

Ok, the database format is compliant with the data model. (Note: the views have not been checked)

SQL commands to copy/paste:

```
ALTER VIEW `view_ActionEmail` AS SELECT DISTINCT `_priv_action_email`.`id` AS `id`,
`_priv_action`.`name` AS `name`, `_priv_action`.`description` AS `description`,
`_priv_action`.`status` AS `status`, `_priv_action_email`.`test_recipient` AS
`test_recipient`, `_priv_action_email`.`from` AS `from`, `_priv_action_email`.`reply_to`
AS `reply_to`, `_priv_action_email`.`to` AS `to`, `_priv_action_email`.`cc` AS `cc`,
`_priv_action_email`.`bcc` AS `bcc`, `_priv_action_email`.`subject` AS `subject`,
`_priv_action_email`.`body` AS `body`, `_priv_action_email`.`importance` AS `importance`,
`_priv_action`.`realclass` AS `finalclass`, CAST(CONCAT(COALESCE(`_priv_action`.`name`,
'')) AS CHAR) AS `friendlyname` FROM `priv_action_email` AS `_priv_action_email`INNER JOIN
(`priv_action` AS `_priv_action` ) ON `_priv_action_email`.`id` = `_priv_action`.`id`
WHERE 1  ;
```

If you navigate to the details of a Server in iTop, you should now see the following:



Modification of Server: Server1

General informations

Name	Server1
Organization	Demo
Status	production
Business criticity	low
Location	Bordeaux
Rack	-- select one --
Enclosure	-- select one --

More informations

Brand	HP
Model	DL380
OS family	Linux
OS version	Unbuntu 11.10
OS licence	-- select one --
CPU	
RAM	
NB U	
Serial number	
Asset number	

Date

Move to production date	
Purchase date	
End of warranty	

Other informations

PowerA source	-- select one --
PowerB source	-- select one --
Description	
Additional Notes	

Final Customization Module

The final result of the customization is available in the zip file below:

sample-add-attribute.zip

Next Steps

You can use the same process to add more fields to the same object, or to alter other objects in iTop.

If you want the added fields to appear either in the default “list” view or “search” form for the modified class of objects, the corresponding “presentation” list must be redefined as well.

To deploy your customization to another iTop server, simply copy the folder “sample-add-attribute” to the extensions folder of iTop and run the setup again.

Creating a new class of CI: Monitor

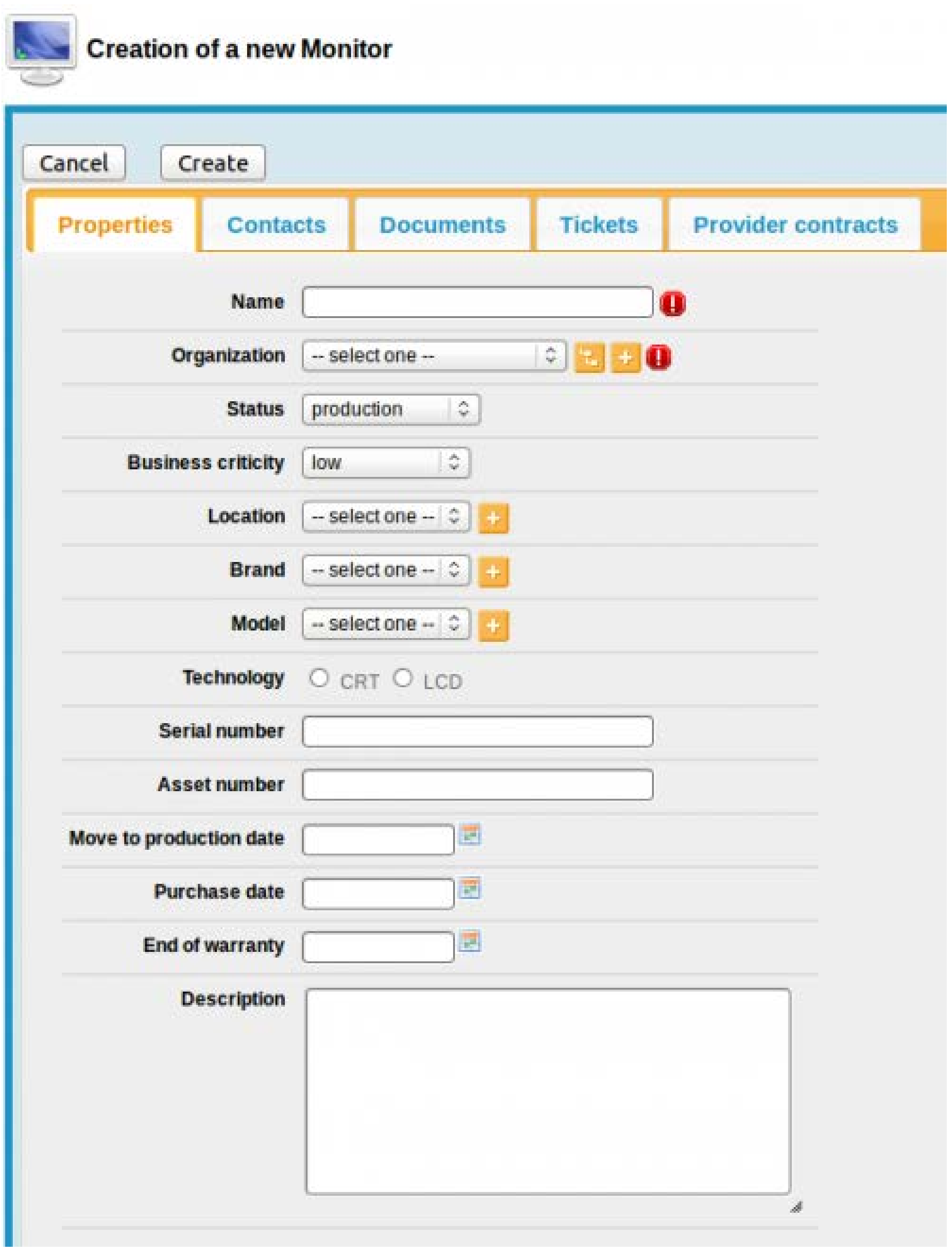
This document explains, step by step, how to create your own iTop module in order to add a new class of CIs: a Monitor.

Goals of this tutorial

In this step-by-step tutorial you will learn to:

- create your own extension module for iTop 2.0
- create a new class of object
- add a new entry into an existing dashboard

For the purpose of this tutorial we will create a new class of CI, called Monitor, that will be very similar to the Peripheral class, with the addition of an extra field “technology” to distinguish CRT and LCD monitors.


Creation of a new Monitor

Name	
Organization	-- select one --
Status	production
Business criticity	low
Location	-- select one --
Brand	-- select one --
Model	-- select one --
Technology	○ CRT ○ LCD
Serial number	
Asset number	
Move to production date	
Purchase date	
End of warranty	
Description	

What you will need

- iTop installed on a development machine, on which you can easily access/edit the files.
- A text editor capable of editing PHP and XML file and supporting UTF-8. On Windows you can use Wordpad (Notepad does not like Unix line endings) or one of the excellent free development IDEs like PSPad or Notepad++.

Customization process

The customization process is the following:

1. Install a development instance of iTop. It is always better not to experiment in production !!
2. Install the toolkit to assist you in the customization
3. Create a new (empty) module using the module creation wizard
4. Copy this new module to the `extensions` folder on iTop and run the setup again to install the empty module
5. Modify the module in `extensions` and use the toolkit to check your customizations

Repeat the last point until you are satisfied with your customization. When you are done, your new module is ready to be deployed. Copy the module folder in the extension directory on your production iTop instance and run the setup to install it.

Step by step tutorial

Create your customization module

Use the module creation wizard. Fill the form with the following values:

Label	Value	Remarks
Module name	sample-add-class	Names starting with `itop-` and `combodo-` are reserved for use by Combodo. It is recommended not to put spaces or accentuated characters in the name of the module. Two modules with the same name cannot co-exist in the same iTop instance.
Module Label	Add Class Sample	This label will be displayed in the setup wizard. Localized characters and spaces are allowed
Module Version	1.0.0	The convention is to use a 3 digits numbering scheme: X.Y.Z
Category	business	Modules that provide modifications to the data model should be in the category 'business'
Dependencies	itop-config-mgmt/2.0.0,'itop-endusers-devices/2.0.0	Our customization module

depends on the modules: iTop Configuration Management (version 2.0.0) in which the Physical Device class is defined and iTop End User Devices (version 2.0.0) in which the "End User Devices" menu is defined

Click **Generate !** to download the empty module as a zip file.

When a module extends an existing class, it **must** be loaded after the module that declared the class to be extended. To achieve this, make sure that the first module is listed in the *dependencies* of your new module.

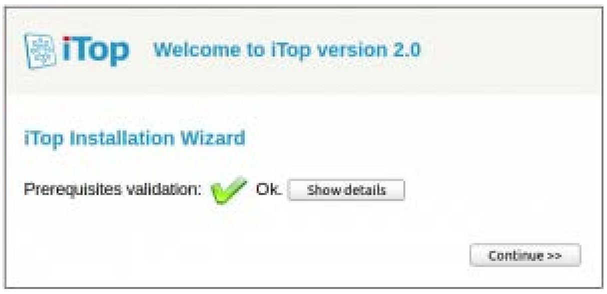
For example if you want to create a new class, dervied from the class VirtualMachine, your custom module must depend on itop-virtualization-mgmt/2.0.0

Install the empty module

Expand the content of the zip into the extensions folder of your development iTop instance. You should now have a folder named sample-add-class inside the extensions folder. this folder contains the following files:

- datamodel.sample-add-class.xml
- module.sample-add-class.php
- en.dict.sample-add-class.php
- model.sample-add-class.php

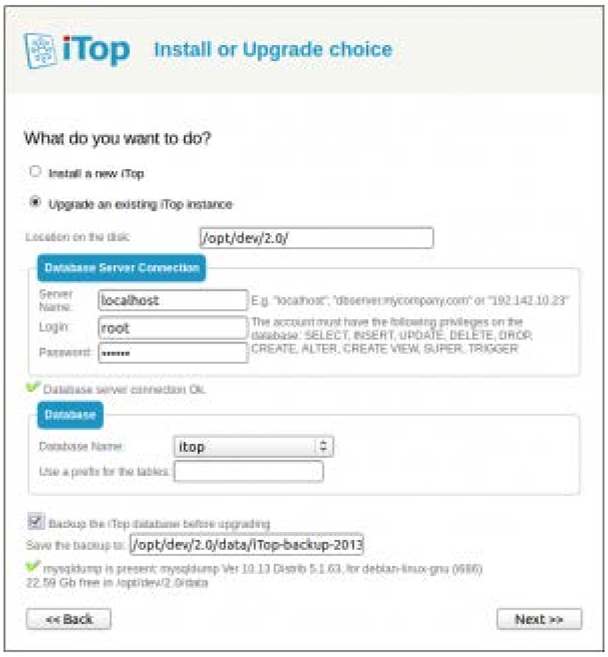
Make sure that the file conf/production/config-itop.php is writable for the web server (on Windows: right click to display the file properties and uncheck the read-only flag; on Linux change the rights of the file), then launch the iTop installation by pointing your browser to **http://your_itop/setup/**


Welcome to iTop version 2.0

iTop Installation Wizard

Prerequisites validation: ✓ Ok

Click “Continue »” to start the re-installation.



What do you want to do?

○ Install a new iTop

● Upgrade an existing iTop instance

Location on the disk: /opt/dev/2.0/

Database Server Connection

Server Name: localhost — E.g. "localhost", "dbserver.mycompany.com" or "192.142.10.23"

Login: root — The account must have the following privileges on the database: SELECT, INSERT, UPDATE, DELETE, DROP, CREATE, ALTER, CREATE VIEW, SUPER, TRIGGER

Password: ••••••

✓ Database server connection Ok.

Database

Database Name: itop

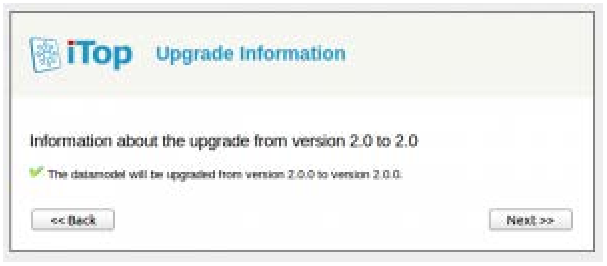
Use a prefix for the tables:

☑ Backup the iTop database before upgrading

Save the backup to: /opt/dev/2.0/data/iTop-backup-2013

✓ mysqldump is present: mysqldump Ver 10.13 Distrib 5.1.63, for debian-linux-gnu (i686)
22.59 Gb free in /opt/dev/2.0/data

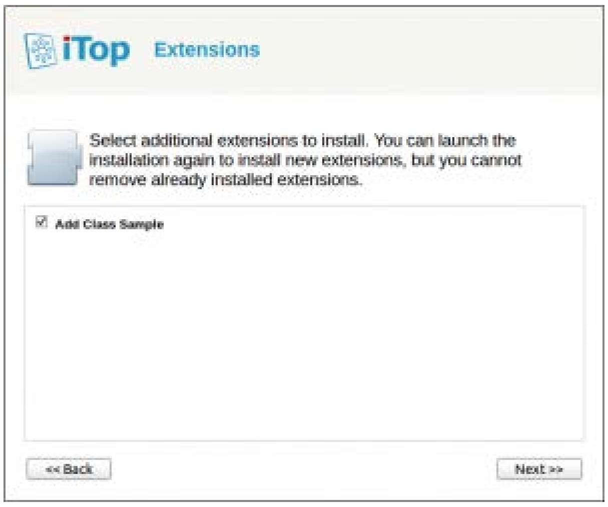
Make sure that “Update an existing instance” is selected before clicking “Next »”.



Information about the upgrade from version 2.0 to 2.0

✓ The datamodel will be upgraded from version 2.0.0 to version 2.0.0.

Continue to the next steps of the wizard…



Select additional extensions to install. You can launch the installation again to install new extensions, but you cannot remove already installed extensions.

☑ Add Class Sample

Your custom module should appear in the list of “Extensions”. If it's not the case, check that the module files were copied in the proper location and that the web server has enough rights to read them.

Select your custom module before clicking “Next »” and complete the installation.

Add the Monitor class

Using you favorite text editor, open the file datamodel.sample-add-class.xml.

Remove the tags <profiles></profiles> since the module will not contain any profile definition.

Inside the classes tag, add the following piece of code:

```
    <class id="Monitor" _delta="define">
      <parent>PhysicalDevice</parent>
      <properties>
        <category>bizmodel,searchable</category>
        <abstract>false</abstract>
        <key_type>autoincrement</key_type>
        <db_table>monitor</db_table>
        <db_key_field>id</db_key_field>
        <db_final_class_field/>
        <naming>
          <format>%1$s</format>
          <attributes>
            <attribute id="name"/>
          </attributes>
        </naming>
        <display_template/>
        <icon>images/monitor.png</icon>
        <reconciliation>
          <attributes>
            <attribute id="name"/>
            <attribute id="org_id"/>
            <attribute id="organization_name"/>
          </attributes>
        </reconciliation>
      </properties>
      <fields>
        <field id="technology" xsi:type="AttributeEnum">
```

```
          <values>
            <value>crt</value>
            <value>lcd</value>
          </values>
          <sql>technology</sql>
          <default_value/>
          <is_null_allowed>true</is_null_allowed>
          <display_style>radio_horizontal</display_style>
        </field>
      </fields>
    </class>
```

This instructs iTop to define a new class derived from PhysicalDevice. An extra table “monitor” will be added in SQL to store the class specific data. The “Monitor” class adds one extra field “technology” to the PhysicalDevice class. This field is an enumerated value with two possible values “lcd” and “crt”. This field is to be stored in the column “technology” in the SQL database.

For more information about the meaning of the various parameters of the class and field tags, refer to the XML reference documentation.

The <presentation> tag must also be defined to describe how the object should be displayed in iTop. There are three “presentations” to define:

- The “details” defines the form used to display and edit an instance of the object
- The “list” defines the default columns to be used for displaying a list of objects of this class
- The “search” defines the available fields in the search form for this class of objects

Add the following piece of code just after the closing </fields> tag:

```
      <presentation>
        <details>
          <items>
            <item id="name">
              <rank>10</rank>
            </item>
            <item id="org_id">
              <rank>20</rank>
            </item>
            <item id="status">
              <rank>30</rank>
            </item>
            <item id="business_criticity">
```

```
  <rank>40</rank>
</item>
<item id="location_id">
  <rank>50</rank>
</item>
<item id="brand_id">
  <rank>60</rank>
</item>
<item id="model_id">
  <rank>70</rank>
</item>
<item id="technology">
  <rank>75</rank>
</item>
<item id="serialnumber">
  <rank>80</rank>
</item>
<item id="asset_number">
  <rank>90</rank>
</item>
<item id="move2production">
  <rank>100</rank>
</item>
<item id="purchase_date">
  <rank>110</rank>
</item>
<item id="end_of_warranty">
  <rank>120</rank>
</item>
<item id="description">
  <rank>130</rank>
</item>
<item id="contacts_list">
  <rank>140</rank>
</item>
<item id="documents_list">
  <rank>150</rank>
</item>
<item id="tickets_list">
  <rank>160</rank>
</item>
<item id="providercontracts_list">
  <rank>170</rank>
</item>
```

```
    <item id="services_list">
      <rank>180</rank>
    </item>
  </items>
</details>
<search>
  <items>
    <item id="name">
      <rank>10</rank>
    </item>
    <item id="org_id">
      <rank>20</rank>
    </item>
    <item id="status">
      <rank>30</rank>
    </item>
    <item id="business_criticity">
      <rank>40</rank>
    </item>
    <item id="location_id">
      <rank>50</rank>
    </item>
    <item id="brand_id">
      <rank>60</rank>
    </item>
    <item id="model_id">
      <rank>70</rank>
    </item>
    <item id="technology">
      <rank>75</rank>
    </item>
    <item id="serialnumber">
      <rank>80</rank>
    </item>
    <item id="asset_number">
      <rank>90</rank>
    </item>
    <item id="move2production">
      <rank>100</rank>
    </item>
    <item id="purchase_date">
      <rank>110</rank>
    </item>
    <item id="end_of_warranty">
```

```
        <rank>120</rank>
      </item>
    </items>
  </search>
  <list>
    <items>
      <item id="org_id">
        <rank>10</rank>
      </item>
      <item id="status">
        <rank>20</rank>
      </item>
      <item id="business_criticity">
        <rank>30</rank>
      </item>
      <item id="location_id">
        <rank>40</rank>
      </item>
      <item id="brand_id">
        <rank>50</rank>
      </item>
      <item id="model_id">
        <rank>60</rank>
      </item>
      <item id="serialnumber">
        <rank>70</rank>
      </item>
    </items>
  </list>
</presentation>
```

You should now have the following XML file:

datamodel.sample-add-class.xml

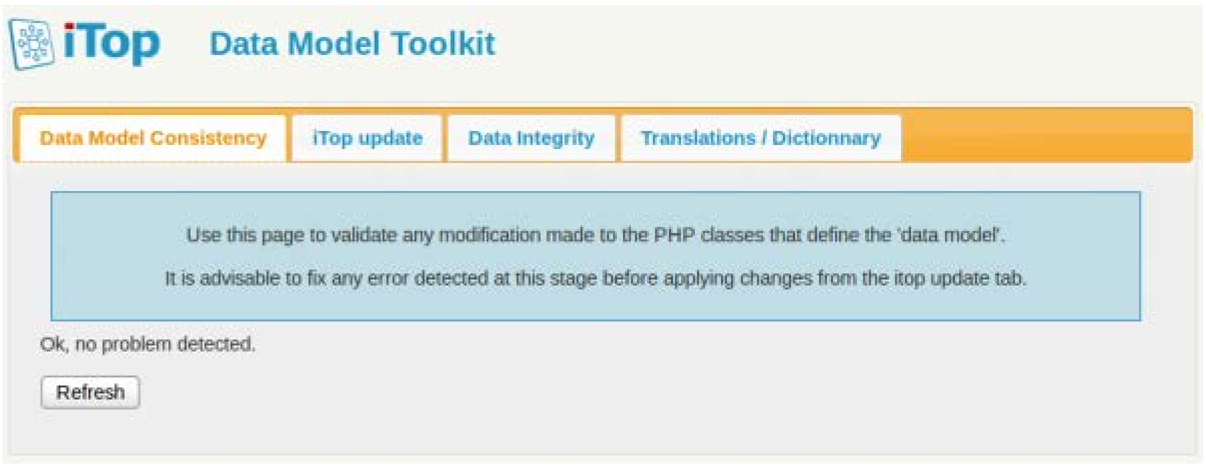
```
<?xml version="1.0" encoding="UTF-8"?>
<itop_design
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.0">
  <classes>
    <class id="Monitor" _delta="define">
      <parent>PhysicalDevice</parent>
      <properties>
        <category>bizmodel,searchable</category>
        <abstract>false</abstract>
  <key_type>autoincrement</key_type>
  <db_table>monitor</db_table>
  <db_key_field>id</db_key_field>
  <db_final_class_field/>
  <naming>
    <format>%1$s</format>
    <attributes>
      <attribute id="name"/>
    </attributes>
  </naming>
  <display_template/>
  <icon>images/monitor.png</icon>
  <reconciliation>
    <attributes>
      <attribute id="name"/>
      <attribute id="org_id"/>
      <attribute id="organization_name"/>
    </attributes>
  </reconciliation>
</properties>
<fields>
  <field id="technology" xsi:type="AttributeEnum">
    <values>
      <value>crt</value>
      <value>lcd</value>
    </values>
    <sql>technology</sql>
    <default_value/>
    <is_null_allowed>true</is_null_allowed>
    <display_style>radio_horizontal</display_style>
  </field>
</fields>
<methods/>
<presentation>
  <details>
    <items>
      <item id="name">
        <rank>10</rank>
      </item>
      <item id="org_id">
        <rank>20</rank>
      </item>
      <item id="status">
        <rank>30</rank>
```

```
</item>
<item id="business_criticity">
  <rank>40</rank>
</item>
<item id="location_id">
  <rank>50</rank>
</item>
<item id="brand_id">
  <rank>60</rank>
</item>
<item id="model_id">
  <rank>70</rank>
</item>
<item id="technology">
  <rank>75</rank>
</item>
<item id="serialnumber">
  <rank>80</rank>
</item>
<item id="asset_number">
  <rank>90</rank>
</item>
<item id="move2production">
  <rank>100</rank>
</item>
<item id="purchase_date">
  <rank>110</rank>
</item>
<item id="end_of_warranty">
  <rank>120</rank>
</item>
<item id="description">
  <rank>130</rank>
</item>
<item id="contacts_list">
  <rank>140</rank>
</item>
<item id="documents_list">
  <rank>150</rank>
</item>
<item id="tickets_list">
  <rank>160</rank>
</item>
<item id="providercontracts_list">
```

```
      <rank>170</rank>
    </item>
    <item id="services_list">
      <rank>180</rank>
    </item>
  </items>
</details>
<search>
  <items>
    <item id="name">
      <rank>10</rank>
    </item>
    <item id="org_id">
      <rank>20</rank>
    </item>
    <item id="status">
      <rank>30</rank>
    </item>
    <item id="business_criticity">
      <rank>40</rank>
    </item>
    <item id="location_id">
      <rank>50</rank>
    </item>
    <item id="brand_id">
      <rank>60</rank>
    </item>
    <item id="model_id">
      <rank>70</rank>
    </item>
    <item id="technology">
      <rank>75</rank>
    </item>
    <item id="serialnumber">
      <rank>80</rank>
    </item>
    <item id="asset_number">
      <rank>90</rank>
    </item>
    <item id="move2production">
      <rank>100</rank>
    </item>
    <item id="purchase_date">
      <rank>110</rank>
```

```
            </item>
            <item id="end_of_warranty">
              <rank>120</rank>
            </item>
          </items>
        </search>
        <list>
          <items>
            <item id="org_id">
              <rank>10</rank>
            </item>
            <item id="status">
              <rank>20</rank>
            </item>
            <item id="business_criticity">
              <rank>30</rank>
            </item>
            <item id="location_id">
              <rank>40</rank>
            </item>
            <item id="brand_id">
              <rank>50</rank>
            </item>
            <item id="model_id">
              <rank>60</rank>
            </item>
            <item id="serialnumber">
              <rank>70</rank>
            </item>
          </items>
        </list>
      </presentation>
    </class>
  </classes>
<menus>
</menus>
</itop_design>
```

Check your modification by running the toolkit. Point your browser to http://your_itop/toolkit.



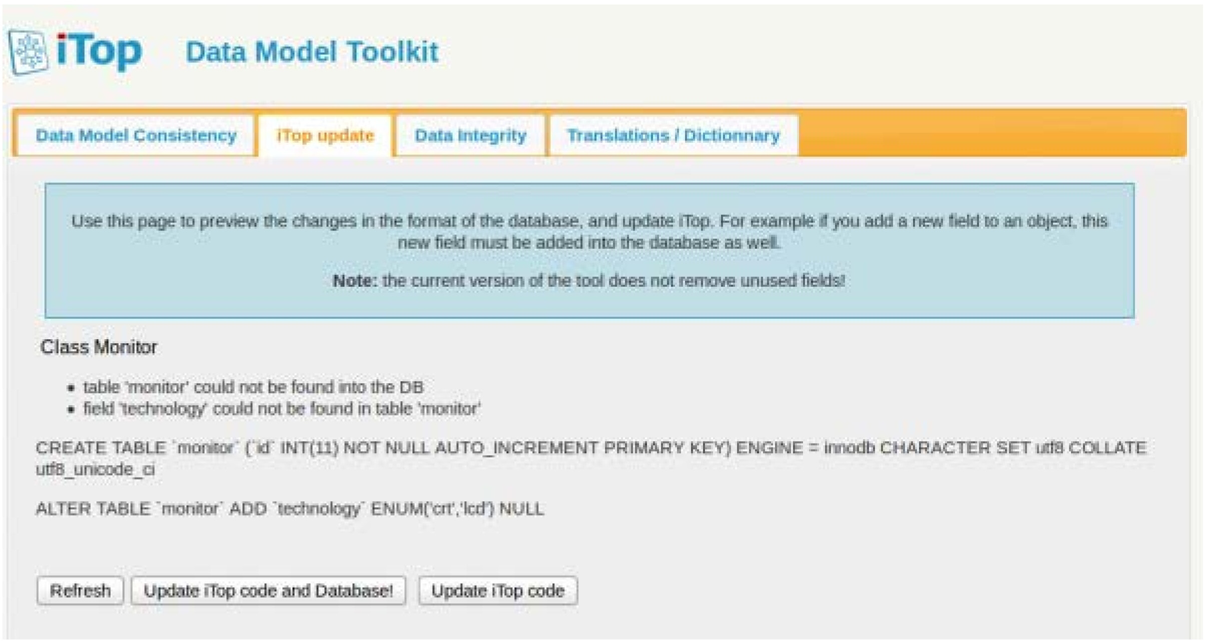
Use this page to validate any modification made to the PHP classes that define the 'data model'.

It is advisable to fix any error detected at this stage before applying changes from the itop update tab.

Ok, no problem detected.

If any error is reported at this stage, fix it by editing the XML file and check again your modifications by clicking on the “Refresh” button in the toolkit page.

Once all the errors have been fixed, you can apply the modifications to iTop by using the second tab of the toolkit:



Use this page to preview the changes in the format of the database, and update iTop. For example if you add a new field to an object, this new field must be added into the database as well.

Note: the current version of the tool does not remove unused fields!

Class Monitor

- table 'monitor' could not be found into the DB
- field 'technology' could not be found in table 'monitor'

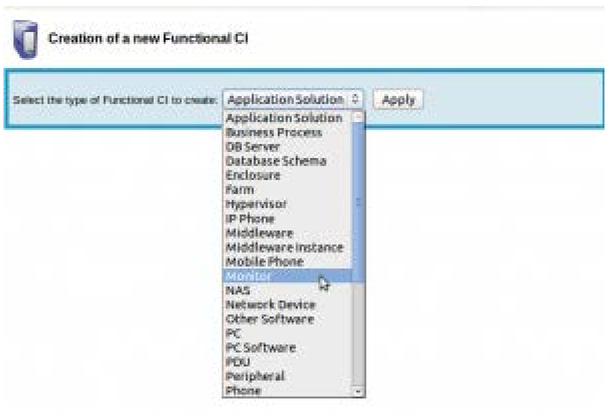
```
CREATE TABLE `monitor` (`id` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY) ENGINE = innodb CHARACTER SET utf8 COLLATE utf8_unicode_ci

ALTER TABLE `monitor` ADD `technology` ENUM('crt','lcd') NULL
```

Click on the button **Update iTop Code and Database!** to:

1. Compile the XML data model to PHP classes
2. Update the database schema (creation of the monitor table).

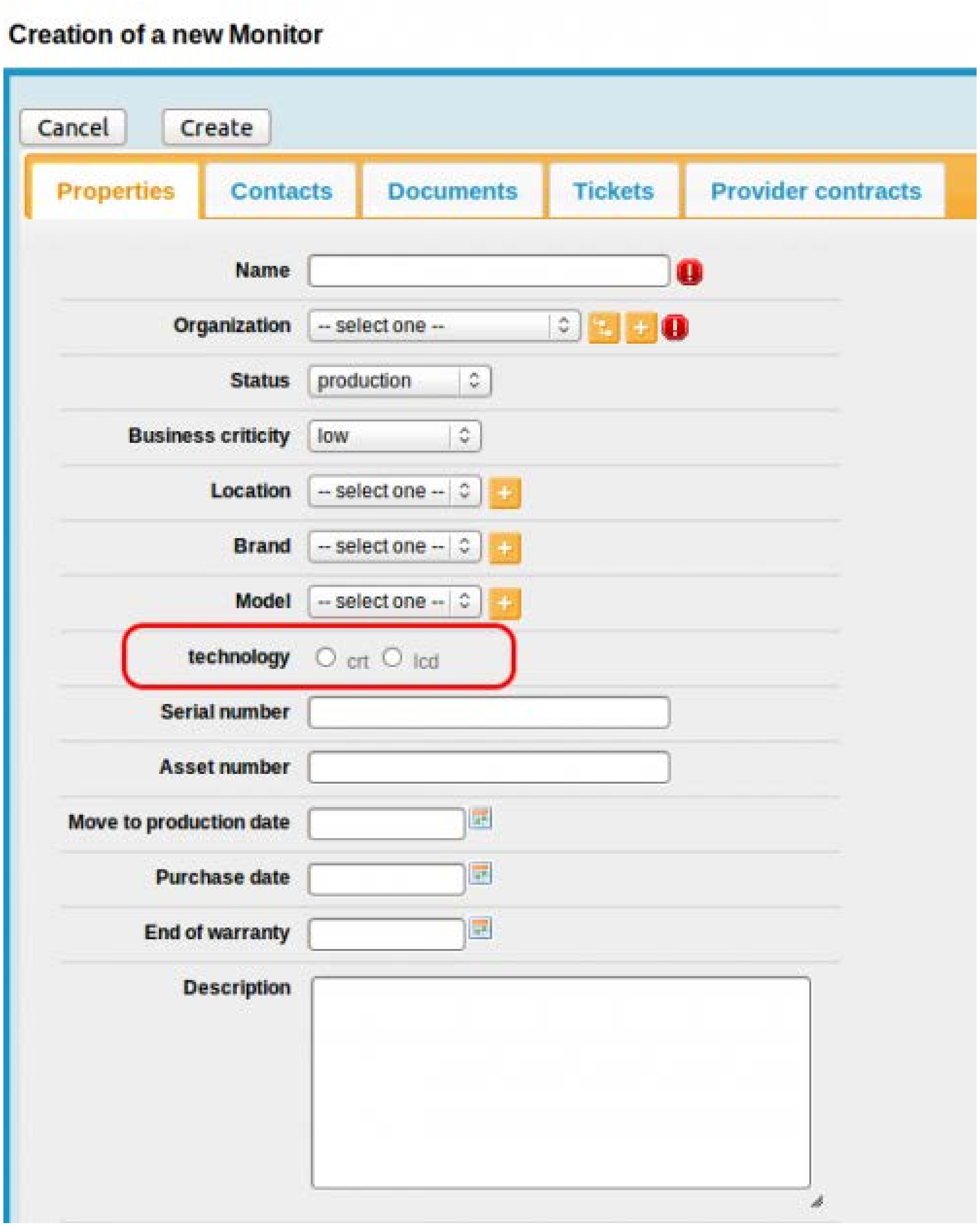
At this point, if you navigate in iTop and click on the “Configuration Management / New CI” menu, you can see that “Monitor” is available in the drop-down list:


Creation of a new Functional CI

Select the type of Functional CI to create: Application Solution | Apply

- Application Solution
- Business Process
- DB Server
- Database Schema
- Enclosure
- Farm
- Hypervisor
- IP Phone
- Middleware
- Middleware Instance
- Mobile Phone
- Monitor
- NAS
- Network Device
- Other Software
- PC
- PC Software
- PDU
- Peripheral
- Phone

The following form gets displayed:


Creation of a new Monitor

Cancel | Create

Properties | Contacts | Documents | Tickets | Provider contracts

Name	
Organization	-- select one --
Status	production
Business criticity	low
Location	-- select one --
Brand	-- select one --
Model	-- select one --
technology	○ crt ○ lcd
Serial number	
Asset number	
Move to production date	
Purchase date	
End of warranty	
Description	

As you can see the new “Monitor” class seems to work fine already. However there are two missing pieces:

- The class has no icon associated with it,
- The label of the “technology” field (and its values) are in lowercase.

Add an Icon for the class

In the datamodel.sample-add-class.xml the class icon is defined as:

```
<icon>images/monitor.png</icon>
```

This means that itop expects a file named monitor.png in the images subfolder of the module.

To provide the icon, create the images folder inside the extensions/sample-add-class folder and copy a nice monitor icon - as monitor.png - inside it.

Icon files can be in any format that is commonly supported by web browsers (JPEG, GIF, PNG…), but PNG-24 is the only format that supports smooth transparency. The recommended size for class icons is 48×48 pixels.
A good source of nice icons is http://www.iconfinder.com/ (pay attention to the licences)

Fix the labels

In order to have the new technology field display properly, you need to edit the *dictionary*.

Using you text editor, open the file en.dict.sample-add-class.php.

Insert the lines:

```
        'Class:Monitor' => 'Monitor',
        'Class:Monitor+' => 'A computer display',
        'Class:Monitor/Attribute:technology' => 'Technology',
        'Class:Monitor/Attribute:technology+' => 'Technology used for
the display',
        'Class:Monitor/Attribute:technology/Value:crt' => 'CRT',
        'Class:Monitor/Attribute:technology/Value:lcd' => 'LCD',
```

Just below the comment:

```
        // Dictionary entries go here
```

The first two lines are the label for the class and a short explanation about the meaning of the class. The other lines provides the translated label for the new field and its values.

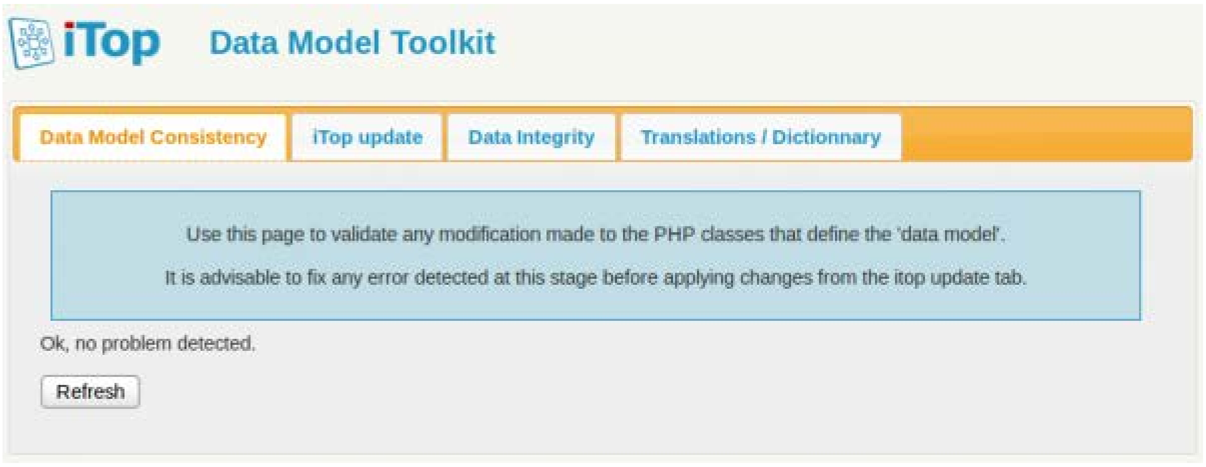
You should obtain the following file:

en.dict.sample-add-class.php

```
<?php
/**
 * Localized data
 *
 * @copyright   Copyright (C) 2013 XXXXX
 * @license     http://opensource.org/licenses/AGPL-3.0
 */

Dict::Add('EN US', 'English', 'English', array(
  // Dictionary entries go here
  'Class:Monitor' => 'Monitor',
  'Class:Monitor+' => 'A computer display',
  'Class:Monitor/Attribute:technology' => 'Technology',
  'Class:Monitor/Attribute:technology+' => 'Technology used for
the display',
  'Class:Monitor/Attribute:technology/Value:crt' => 'CRT',
  'Class:Monitor/Attribute:technology/Value:lcd' => 'LCD',
));
?>
```

One more time, check your modification by running the toolkit.



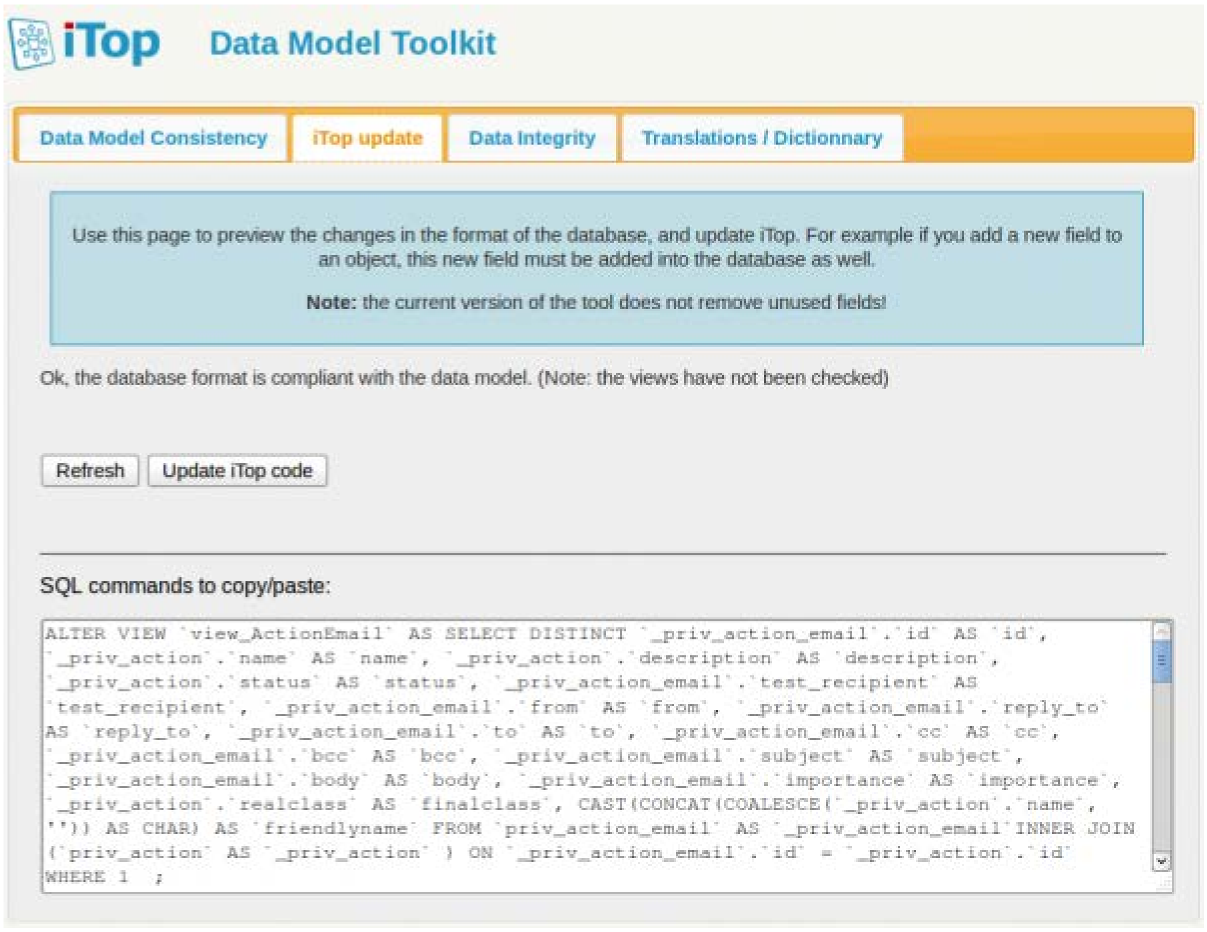
Use this page to validate any modification made to the PHP classes that define the 'data model'.

It is advisable to fix any error detected at this stage before applying changes from the itop update tab.

Ok, no problem detected.

If errors are reported at this stage, fix them by editing the PHP file and check again your modifications by clicking on the "Refresh" button in the toolkit page.

Once all the errors have been fixed, you can apply the modifications to iTop by using "Update iTop Code" button on the second tab of the toolkit:



Use this page to preview the changes in the format of the database, and update iTop. For example if you add a new field to an object, this new field must be added into the database as well.

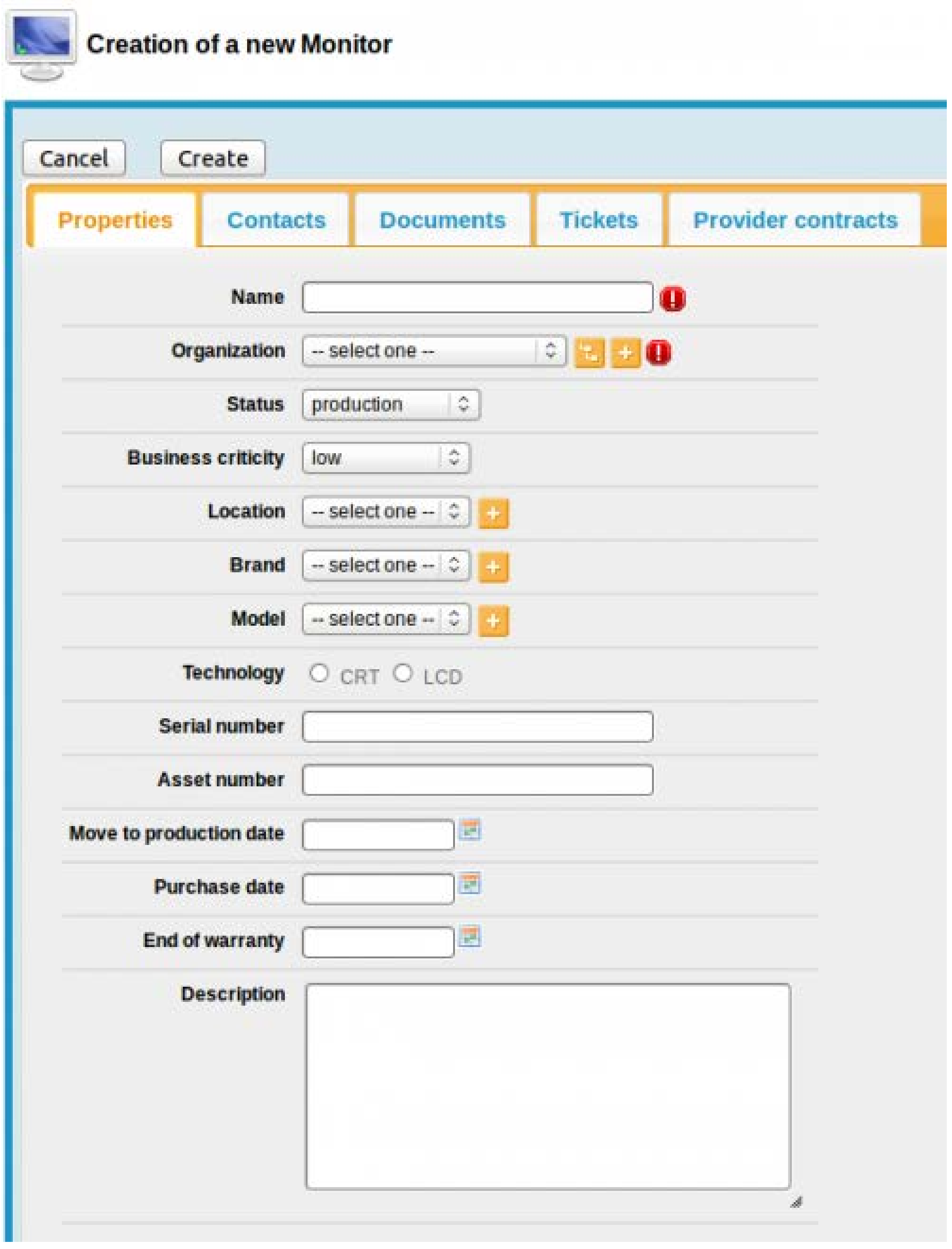
Note: the current version of the tool does not remove unused fields!

Ok, the database format is compliant with the data model. (Note: the views have not been checked)

SQL commands to copy/paste:

```
ALTER VIEW `view_ActionEmail` AS SELECT DISTINCT `_priv_action_email`.`id` AS `id`,
`_priv_action`.`name` AS `name`, `_priv_action`.`description` AS `description`,
`_priv_action`.`status` AS `status`, `_priv_action_email`.`test_recipient` AS
`test_recipient`, `_priv_action_email`.`from` AS `from`, `_priv_action_email`.`reply_to`
AS `reply_to`, `_priv_action_email`.`to` AS `to`, `_priv_action_email`.`cc` AS `cc`,
`_priv_action_email`.`bcc` AS `bcc`, `_priv_action_email`.`subject` AS `subject`,
`_priv_action_email`.`body` AS `body`, `_priv_action_email`.`importance` AS `importance`,
`_priv_action`.`realclass` AS `finalclass`, CAST(CONCAT(COALESCE(`_priv_action`.`name`,
'')) AS CHAR) AS `friendlyname` FROM `priv_action_email` AS `_priv_action_email`INNER JOIN
(`priv_action` AS `_priv_action` ) ON `_priv_action_email`.`id` = `_priv_action`.`id`
WHERE 1  ;
```

If you navigate to the details of a Server in iTop, you should now see the following:


Creation of a new Monitor

Name	
Organization	-- select one --
Status	production
Business criticity	low
Location	-- select one --
Brand	-- select one --
Model	-- select one --
Technology	○ CRT ○ LCD
Serial number	
Asset number	
Move to production date	
Purchase date	
End of warranty	
Description	

Add a dashboard item

At this stage, the only way to access the Monitors in iTop is either by searching for any CI, or through the "New CI" menu.

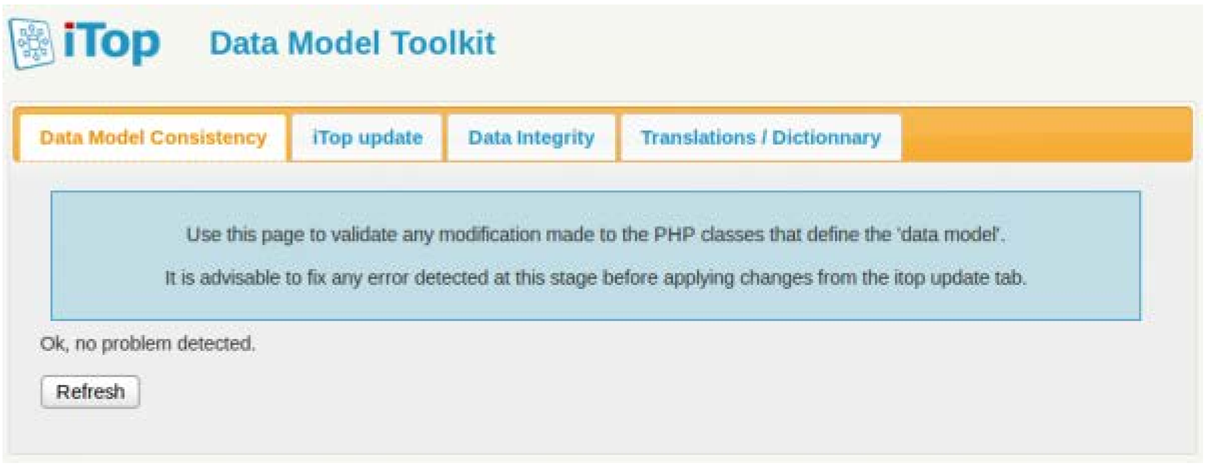
Let's add the Monitor object to the "Configuration Management / Overview" dashboard, in the "End User Devices" section.

In order to provide the expected result, our customization module will have to alter the definition of this dashboard. This can be achieved by replacing the empty <menus></menus> tags by the following XML definition in the file datamodel.sample-add-class.xml:

```
<menus>
  <menu id="ConfigManagementOverview" xsi:type="DashboardMenuNode"
_delta="must_exist">
    <definition>
      <cells>
        <cell id="2" _delta="must_exist">
          <dashlets>
            <dashlet id="99" xsi:type="DashletBadge" _delta="define">
              <rank>8</rank>
              <class>Monitor</class>
            </dashlet>
          </dashlets>
        </cell>
      </cells>
    </definition>
  </menu>
</menus>
```

The Overview dashboard is defined in several parts in the iTop data model. The 'End User Devices" section of the dashboard is defined by the "itop-enduser-devices" module. In order to be able to alter this definition, our customization module must be declared as depending on "itop-enduser-devices", so that it will be loaded after this module.

One more time, check your modification by running the toolkit.



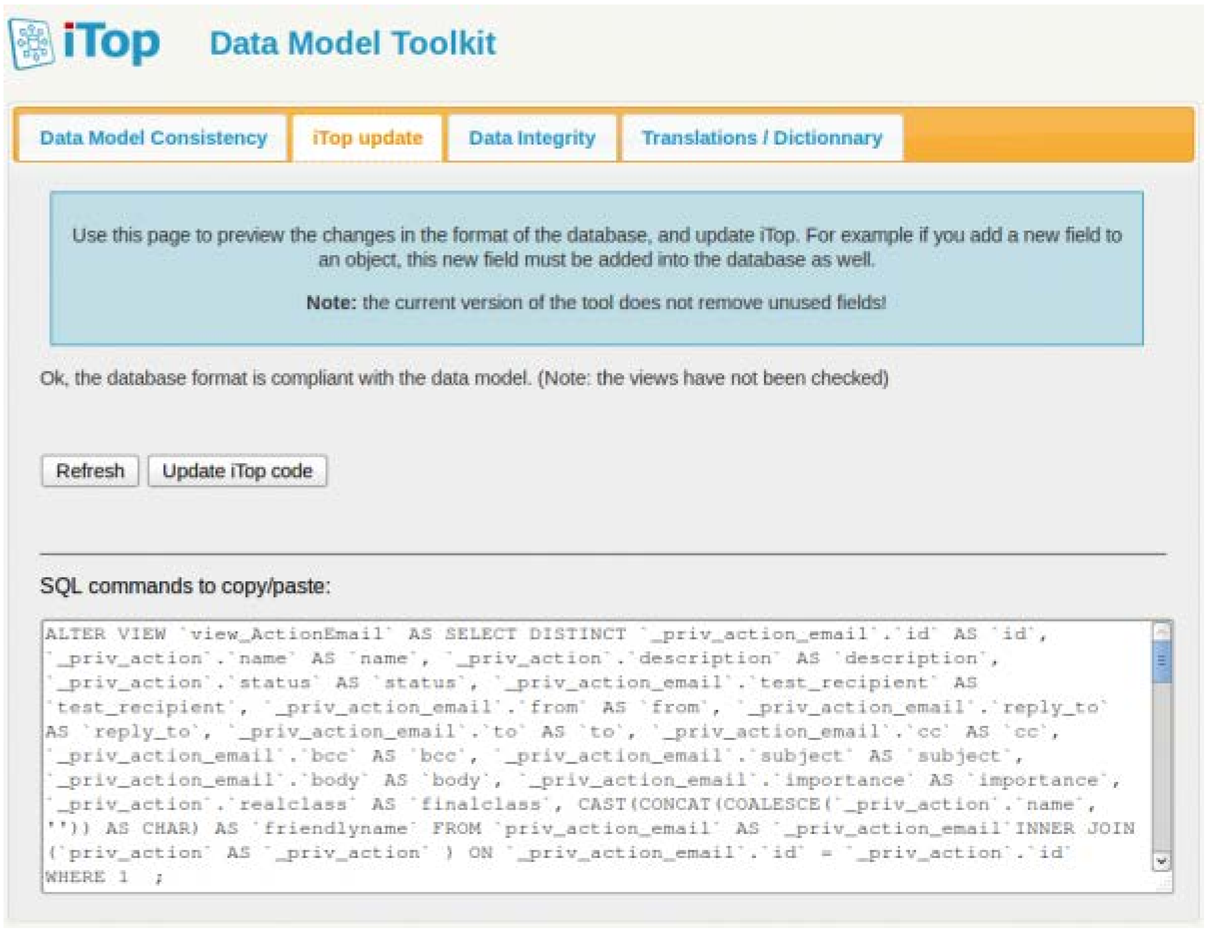
Use this page to validate any modification made to the PHP classes that define the 'data model'.

It is advisable to fix any error detected at this stage before applying changes from the itop update tab.

Ok, no problem detected.

If errors are reported at this stage, fix them by editing the PHP file and check again your modifications by clicking on the "Refresh" button in the toolkit page.

Once all the errors have been fixed, you can apply the modifications to iTop by using “Update iTop Code” button on the second tab of the toolkit:



Use this page to preview the changes in the format of the database, and update iTop. For example if you add a new field to an object, this new field must be added into the database as well.

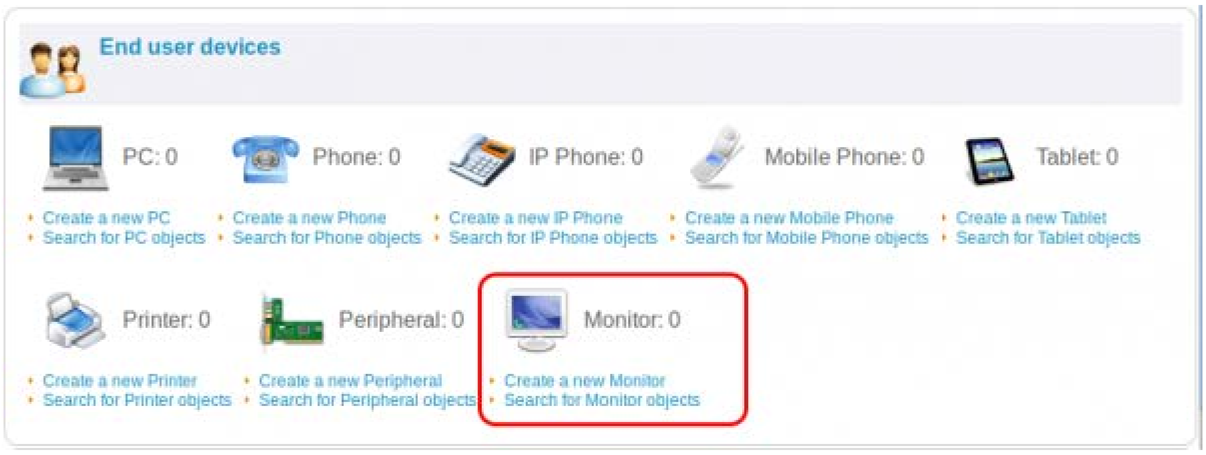
Note: the current version of the tool does not remove unused fields!

Ok, the database format is compliant with the data model. (Note: the views have not been checked)

SQL commands to copy/paste:

```
ALTER VIEW `view_ActionEmail` AS SELECT DISTINCT `_priv_action_email`.`id` AS `id`,
`_priv_action`.`name` AS `name`, `_priv_action`.`description` AS `description`,
`_priv_action`.`status` AS `status`, `_priv_action_email`.`test_recipient` AS
`test_recipient`, `_priv_action_email`.`from` AS `from`, `_priv_action_email`.`reply_to`
AS `reply_to`, `_priv_action_email`.`to` AS `to`, `_priv_action_email`.`cc` AS `cc`,
`_priv_action_email`.`bcc` AS `bcc`, `_priv_action_email`.`subject` AS `subject`,
`_priv_action_email`.`body` AS `body`, `_priv_action_email`.`importance` AS `importance`,
`_priv_action`.`realclass` AS `finalclass`, CAST(CONCAT(COALESCE(`_priv_action`.`name`,
'')) AS CHAR) AS `friendlyname` FROM `priv_action_email` AS `_priv_action_email` INNER JOIN
(`priv_action` AS `_priv_action` ) ON `_priv_action_email`.`id` = `_priv_action`.`id`
WHERE 1  ;
```

If you navigate to the “Configuration Management / Overview” menu in iTop, you should now see the following:



Final Customization Module

You can download the complete customization module by clicking on the link below:

sample-add-class.zip

Next Steps

To deploy your customization to another iTop server, simply copy the folder “sample-add-class” to the extensions folder of iTop and run the setup again.

Creating new Profiles

This document explains, step by step, how to create your own iTop module in order to create new profiles to grant access to the iTop application.

Goals of this tutorial

In this step-by-step tutorial you will learn to:

- create your own extension module for iTop 2.0
- define new profiles for iTop
- on-board the new profiles by running the setup again

For the purpose of this tutorial we will create two new profiles:

- A complete read-only profile, which grants the users the rights to browse through the application, but not to change anything in iTop
- A read-only profile similar to the “Portal user” profile which grants the users enough rights to browse through the normal iTop application in read-only mode for most classes but also to use the Portal for submitting User Requests.

What you will need

- iTop installed on a development machine, on which you can easily access/edit the files.
- A text editor capable of editing PHP and XML file and supporting UTF-8. On Windows you can use Wordpad (Notepad does not like Unix line endings) or one of the excellent free development IDEs like PSPad or Notepad++.

Customization process

The customization process is the following:

1. Install a development instance of iTop. It is always better not to experiment in production !!
2. Install the toolkit to assist you in the customization
3. Create a new (empty) module using the module creation wizard
4. Copy this new module to the `extensions` folder on iTop and run the setup again to install the empty module
5. Modify the module in `extensions` and use the toolkit to check your customizations
6. Run the setup again to create the new profile(s)

Repeat the last two points until you are satisfied with your customization. When you are done, your new module is ready to be deployed. Copy the module folder in the extensions directory on your production iTop instance and run the setup to install it.

Step by step tutorial

Create your customization module

Use the module creation wizard. Fill the form with the following values:

Label	Value	Remarks
Module name	sample-add-profile	Names starting with `itop-` and `combodo-` are reserved for use by Combodo. It is recommended not to put spaces or accentuated characters in the name of the module. Two modules with the same name cannot co-exist in the same iTop instance.
Module Label	Add Profile Sample	This label will be displayed in the setup wizard. Localized characters and spaces are allowed
Module Version	1.0.0	The convention is to use a 3 digits numbering scheme: X.Y.Z
Category	business	Modules that provide modifications to the data model should be in the category 'business'
Dependencies	itop-profiles-itil/1.0.0	Our customization module depends on the modules: iTop Profiles ITIL since we will be using the groups defined in this module. Note that this module retained the version 1.0.0 even in iTop 2.0 !!

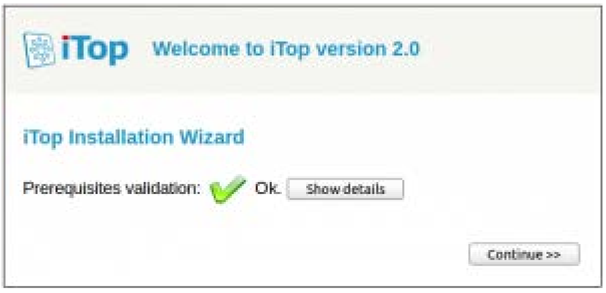
Click **Generate !** to download the empty module as a zip file.

Install the empty module

Expand the content of the zip into the extensions folder of your development iTop instance. You should now have a folder named sample-profile-class inside the extensions folder. this folder contains the following files:

- datamodel.sample-add-profile.xml
- module.sample-add-profile.php
- en.dict.sample-add-profile.php
- model.sample-add-profile.php

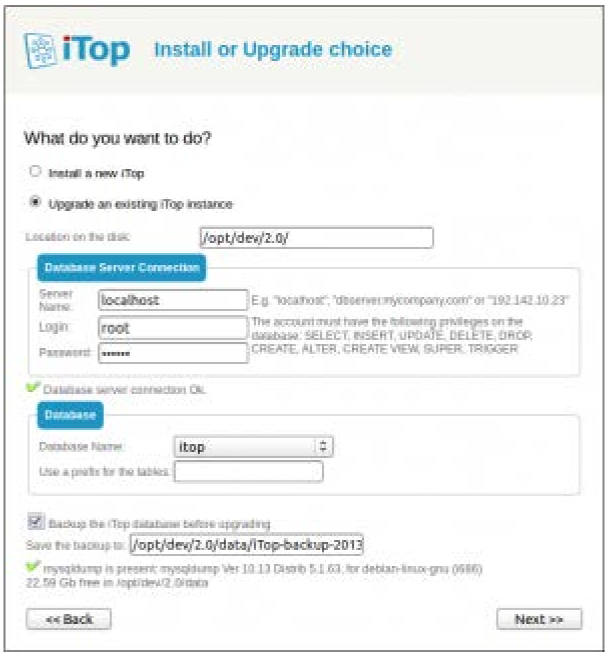
Make sure that the file conf/production/config-itop.php is writable for the web server (on Windows: right click to display the file properties and uncheck the read-only flag; on Linux change the rights of the file), then launch the iTop installation by pointing your browser to http://your_itop/setup/



iTop Installation Wizard

Prerequisites validation: ✓ Ok. Show details

Click "Continue »" to start the re-installation.



What do you want to do?

- ○ Install a new iTop
- ● Upgrade an existing iTop instance

Location on the disk: /opt/dev/2.0/

Database Server Connection

Server Name: localhost E.g. "localhost", "dbserver.mycompany.com" or "192.142.10.23"

Login: root The account must have the following privileges on the database: SELECT, INSERT, UPDATE, DELETE, DROP, CREATE, ALTER, CREATE VIEW, SUPER, TRIGGER

Password: ••••••

✓ Database server connection Ok.

Database

Database Name: itop

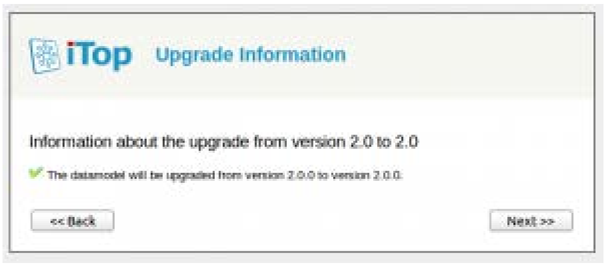
Use a prefix for the tables:

☑ Backup the iTop database before upgrading

Save the backup to: /opt/dev/2.0/data/iTop-backup-2013

✓ mysqldump is present: mysqldump Ver 10.13 Distrib 5.1.63, for debian-linux-gnu (i686)
22.59 Gb free in /opt/dev/2.0/data

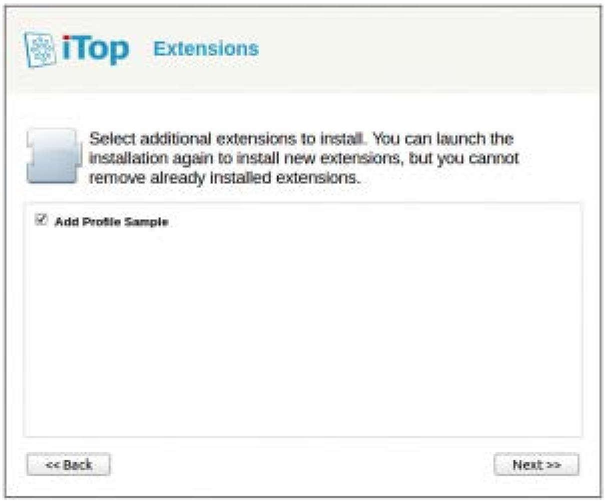
Make sure that “Update an existing instance” is selected before clicking “Next »”.


Upgrade Information

Information about the upgrade from version 2.0 to 2.0

✓ The datamodel will be upgraded from version 2.0.0 to version 2.0.0.

Continue to the next steps of the wizard…


Extensions

Select additional extensions to install. You can launch the installation again to install new extensions, but you cannot remove already installed extensions.

☑ Add Profile Sample

Your custom module should appear in the list of “Extensions”. If this is not the case, check that the module files have been copied in the proper location and that the web server has enough rights to read them.

Select your custom module before clicking “Next »” and complete the installation.

Declare the new Profiles

Using you favorite text editor, open the file datamodel.sample-add-profile.xml.

Inside the user_rights tag, add the following piece of XML:

```
<profiles>
 <profile id="50" _delta="define">
   <name>Read-Only Except Requests</name>
```

```
        <description>Users with this profile are allowed to browse
through all objects in the application and to create/modify user requests
(either through the portal or in the normal application)</description>
        <groups>
          <group id="Portal user - write">
            <actions>
              <action xsi:type="write">allow</action>
            </actions>
          </group>
          <group id="Portal user - delete">
            <actions>
              <action xsi:type="delete">allow</action>
            </actions>
          </group>
          <group id="class:UserRequest">
            <actions>
              <action id="ev_close" xsi:type="stimulus">allow</action>
            </actions>
          </group>
          <group id="*">
            <actions>
              <action xsi:type="read">allow</action>
              <action xsi:type="bulk read">allow</action>
            </actions>
          </group>
        </groups>
      </profile>
      <profile id="51" _delta="define">
        <name>Read-Only No Portal Access</name>
        <description>Users with this profile are allowed to browse
through all objects in the application but not to modify anything (event
through the portal)</description>
        <groups>
          <group id="*">
            <actions>
              <action xsi:type="read">allow</action>
              <action xsi:type="bulk read">allow</action>
            </actions>
          </group>
        </groups>
      </profile>
    </profiles>
```

This instructs iTop to define two new profiles.

- The first profile (numbered id="50") is actually a clone of the "Portal User" profile. The only difference is that "Portal User" is a conventional name for a profile. Any user which has the "Portal User" profile is automatically directed to the portal interface of iTop. Since our new profile is named "Read-Only Except Requests", users with this profile are allowed to navigate through the standard user interface of iTop.
- The second profile (numbered id="51") is a pure read-only profile: it allows only to browse through iTop but not to change anything.

The profiles are defined by accumulating rights on a given set of classes - listed in "groups". By convention the group with id= "*" means "any class". The other groups used in this example are the groups already defined in the module "itop-profiles-itil" (you can see their definition in the file datamodel.itop-profiles-itil.xml).

For example the group "Portal user - write" is defined as follows:

```
<group id="Portal user - write" _delta="define">
  <classes>
    <class id="FileDoc"/>
    <class id="lnkTicketToDoc"/>
    <class id="UserRequest"/>
  </classes>
```

This group is used to grant rights on the classes: FileDoc (a file document), UserRequest (a user request ticket) and also lnkTicketToDoc (the n:n relation between a Document and a Ticket). In order to let the end-user create a User Request ticket (and attach/detach documents to the ticket), the profile "Read-Only Except Requests" must grant write access to all classes in this group (The read access is granted by the rule on the "*" group).

Refer to the XML reference documentation for more information about the XML syntax for groups and profiles.

Since we don't need to redefine any group of classes, the datamodel.add-profile-sample.xml file should contain only the following:

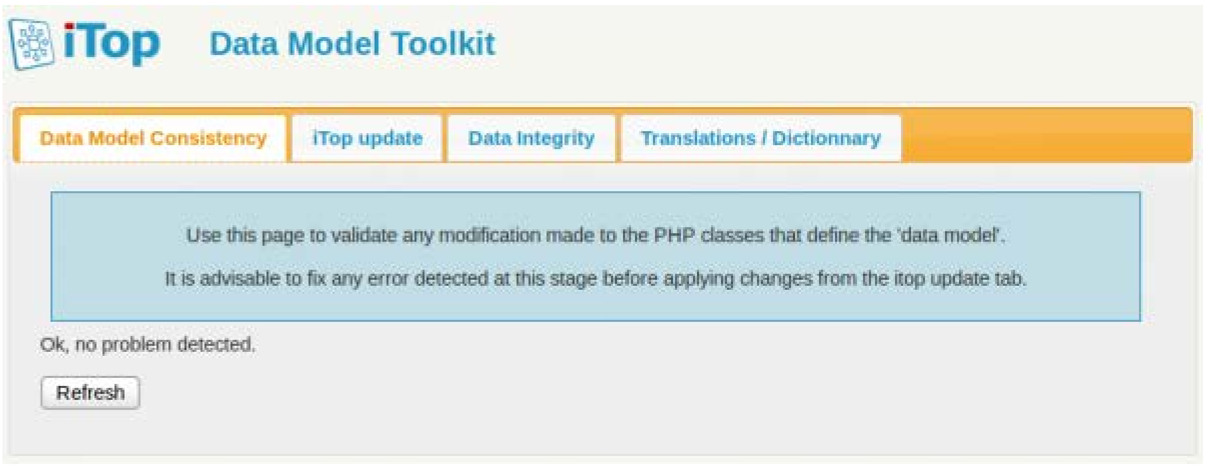
datamodel.sample-add-profile.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<itop_design
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="1.0">
  <classes/>
  <menus/>
  <user_rights>
```

```
     <profiles>
      <profile id="50" _delta="define">
        <name>Read-Only Except Requests</name>
        <description>Users with this profile are allowed to browse
through all objects in the application and to create/modify user
requests (either through the portal or in the normal
application)</description>
        <groups>
          <group id="Portal user - write">
            <actions>
              <action xsi:type="write">allow</action>
            </actions>
          </group>
          <group id="Portal user - delete">
            <actions>
              <action xsi:type="delete">allow</action>
            </actions>
          </group>
          <group id="class:UserRequest">
            <actions>
              <action id="ev_close"
xsi:type="stimulus">allow</action>
            </actions>
          </group>
          <group id="*">
            <actions>
              <action xsi:type="read">allow</action>
              <action xsi:type="bulk read">allow</action>
            </actions>
          </group>
        </groups>
      </profile>
      <profile id="51" _delta="define">
        <name>Read-Only No Portal Access</name>
        <description>Users with this profile are allowed to browse
through all objects in the application but not to modify anything
(event through the portal)</description>
        <groups>
          <group id="*">
            <actions>
              <action xsi:type="read">allow</action>
              <action xsi:type="bulk read">allow</action>
            </actions>
          </group>
```

```
            </groups>
          </profile>
        </profiles>
      </user_rights>
    </itop_design>
```

Check your modification by running the toolkit. Point your browser to **http://your_itop/toolkit**.



Use this page to validate any modification made to the PHP classes that define the 'data model'.

It is advisable to fix any error detected at this stage before applying changes from the itop update tab.

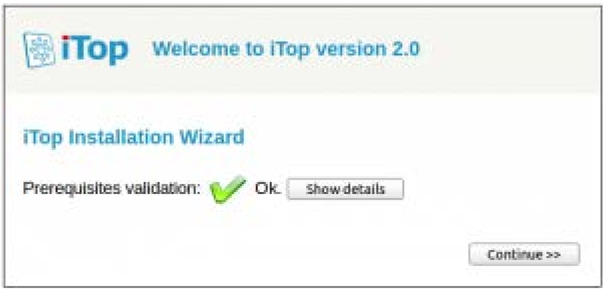
Ok, no problem detected.

If any error is reported at this stage, fix it by editing the XML file and check again your modifications by clicking on the "Refresh" button in the toolkit page.

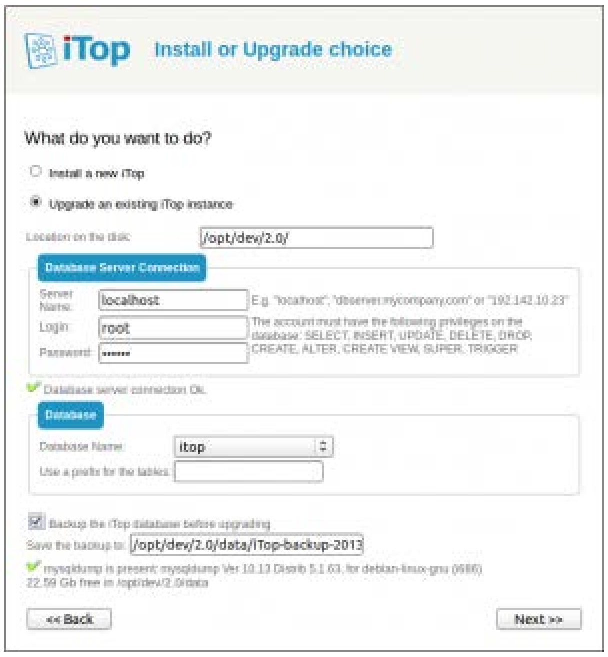
On-board the new Profiles

When you are done with the modifications, you need to **run the setup again** in order to onboard the new profiles.

Make sure that the file conf/production/config-itop.php is writable for the web server (on Windows: right click to display the file properties and uncheck the read-only flag; on Linux change the rights of the file), then launch the iTop installation by pointing your browser to **http://your_itop/setup/**



Click “Continue »” to start the re-installation.



Install or Upgrade choice

What do you want to do?

Install a new iTop [✓ Upgrade an existing iTop instance]

Location on the disk: /opt/dev/2.0/

Database Server Connection

Server Name:	localhost	E.g. "localhost", "dbserver.mycompany.com" or "192.142.10.23"
Login:	root	The account must have the following privileges on the database: SELECT, INSERT, UPDATE, DELETE, DROP, CREATE, ALTER, CREATE VIEW, SUPER, TRIGGER
Password:	••••••	

✓ Database server connection Ok.

Database

Database Name: itop

Use a prefix for the tables:

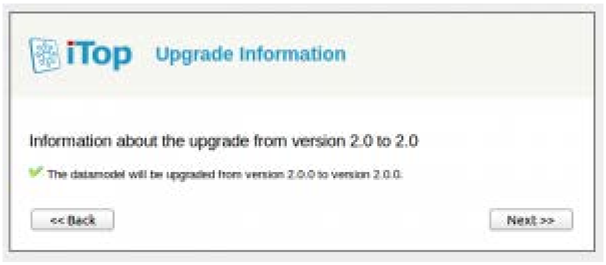
[✓] Backup the iTop database before upgrading

Save the backup to: /opt/dev/2.0/data/iTop-backup-2013

✓ mysqldump is present: mysqldump Ver 10.13 Distrib 5.1.63, for debian-linux-gnu (i686)
22.59 Gb free in /opt/dev/2.0/data

<< Back Next >>

Make sure that “Update an existing instance” is selected before clicking “Next »”.



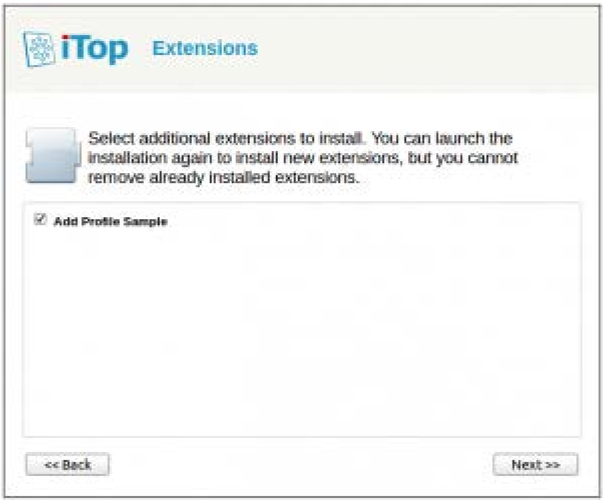
Upgrade Information

Information about the upgrade from version 2.0 to 2.0

✓ The datamodel will be upgraded from version 2.0.0 to version 2.0.0.

<< Back Next >>

Continue to the next steps of the wizard…



Extensions

Select additional extensions to install. You can launch the installation again to install new extensions, but you cannot remove already installed extensions.

[✓] Add Profile Sample

<< Back Next >>

Your custom module should appear in the list of "Extensions", it should already be checked and greyed out (meaning that you cannot deinstall it). Just press "Next »" and complete the installation.

The profiles are defined in the XML but are actually stored in the database. The on-boarding operation that loads them into the database is currently performed only by the setup, so you need to run the setup again each time a new profile is defined (or if a profile definition is modified).

Final Customization Module

You can download the complete customization module by clicking on the link below:

sample-add-profile.zip

Next Steps

To deploy your customization to another iTop server, simply copy the folder "sample-add-profile" to the extensions folder of iTop and run the setup again.