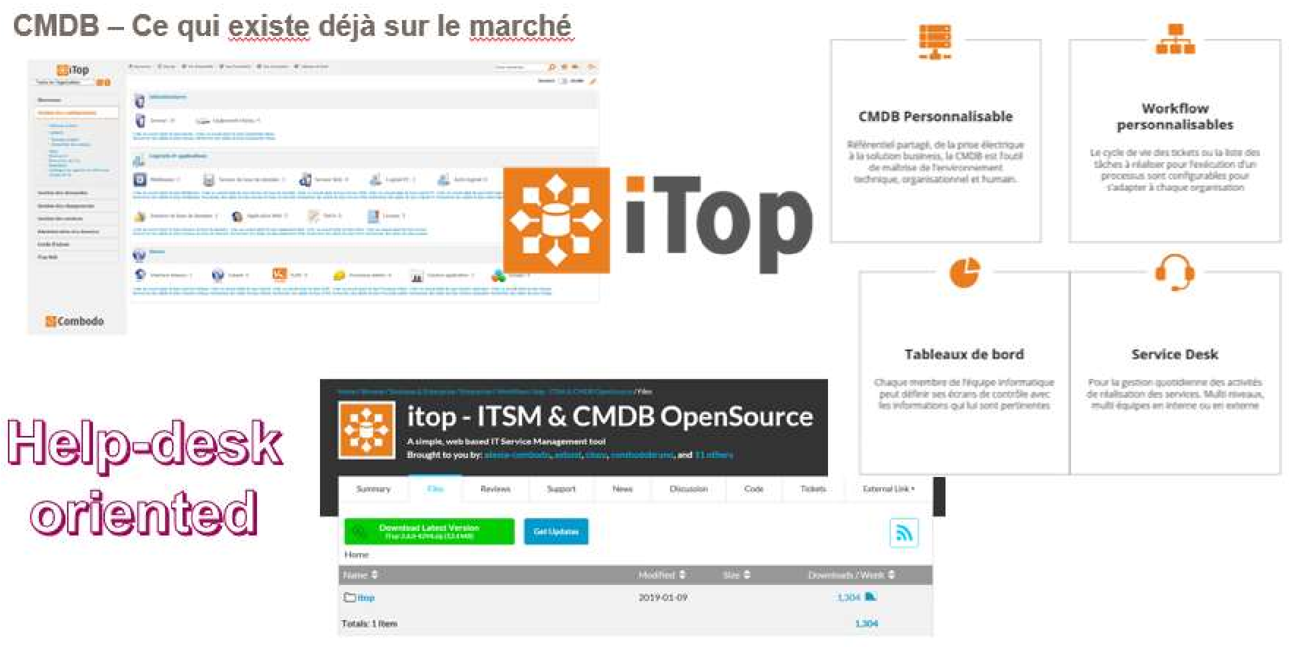
Annexe A
Comparaison des 2 outils



CMDB – Ce qui existe déjà sur le marché

CMDB Personnalisable

Référentiel partagé, de la prise électrique à la solution business, la CMDB est l'outil de maîtrise de l'environnement technique, organisationnel et humain.

Workflow personnalisables

Le cycle de vie des tickets ou la liste des tâches à réaliser pour l'exécution d'un processus sont configurables pour s'adapter à chaque organisation

Tableaux de bord

Chaque membre de l'équipe informatique peut définir ses écrans de contrôle avec les informations qui lui sont pertinentes

Service Desk

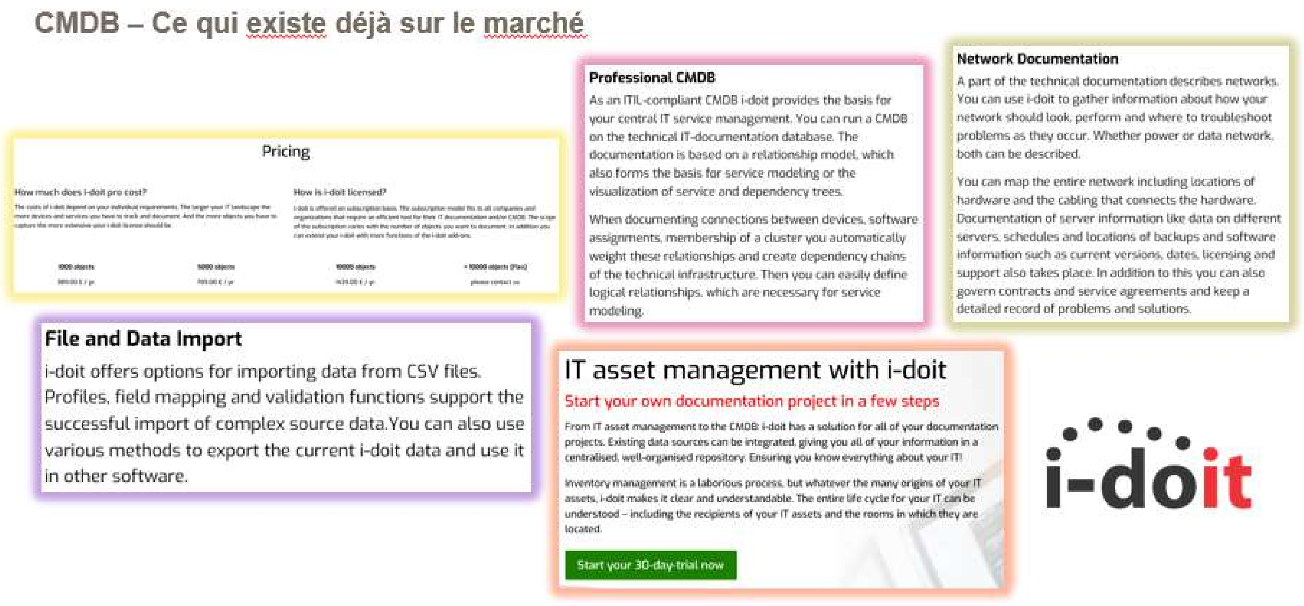
Pour la gestion quotidienne des activités de réalisation des services. Multi niveaux, multi équipes en interne ou en externe

Help-desk oriented

itop - ITSM & CMDB OpenSource

A simple, web based IT Service Management tool

Name	Modified	Size	Downloads / Week
itop	2019-01-09		1,304
Totals: 1 Item			1,304



CMDB – Ce qui existe déjà sur le marché

Pricing

How much does i-doit pro cost?

The costs of i-doit depend on your individual requirements. The larger your IT landscape the more devices and services you have to track and document. And the more objects you have to capture the more extensive your i-doit license should be.

How is i-doit licensed?

i-doit is offered on subscription basis. The subscription model fits to all companies and organisations that require an efficient tool for their IT documentation and/or CMDB. The scope of the subscription varies with the number of objects you want to document. In addition you can extend your i-doit with more functions of the i-doit add-ons.

1000 objects	5000 objects	10000 objects	> 10000 objects (Flex)
389.00 € / yr	709.00 € / yr	1439.00 € / yr	please contact us

Professional CMDB

As an ITIL-compliant CMDB i-doit provides the basis for your central IT service management. You can run a CMDB on the technical IT-documentation database. The documentation is based on a relationship model, which also forms the basis for service modeling or the visualization of service and dependency trees.

When documenting connections between devices, software assignments, membership of a cluster you automatically weight these relationships and create dependency chains of the technical infrastructure. Then you can easily define logical relationships, which are necessary for service modeling.

Network Documentation

A part of the technical documentation describes networks. You can use i-doit to gather information about how your network should look, perform and where to troubleshoot problems as they occur. Whether power or data network, both can be described.

You can map the entire network including locations of hardware and the cabling that connects the hardware. Documentation of server information like data on different servers, schedules and locations of backups and software information such as current versions, dates, licensing and support also takes place. In addition to this you can also govern contracts and service agreements and keep a detailed record of problems and solutions.

File and Data Import

i-doit offers options for importing data from CSV files. Profiles, field mapping and validation functions support the successful import of complex source data. You can also use various methods to export the current i-doit data and use it in other software.

IT asset management with i-doit

Start your own documentation project in a few steps

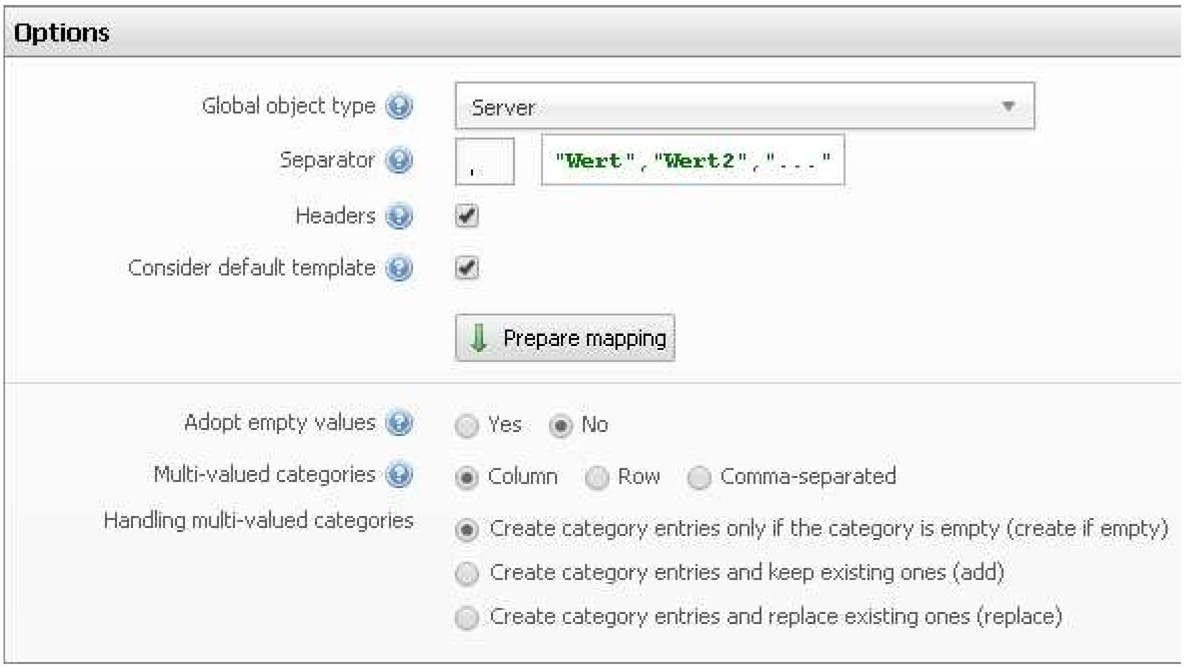
From IT asset management to the CMDB: i-doit has a solution for all of your documentation projects. Existing data sources can be integrated, giving you all of your information in a centralised, well-organised repository. Ensuring you know everything about your IT!

Inventory management is a laborious process, but whatever the many origins of your IT assets, i-doit makes it clear and understandable. The entire life cycle for your IT can be understood – including the recipients of your IT assets and the rooms in which they are located.

Annexe B
Envoi des informations sur Icinga

```
def push_to_icinga(self,ip,hostname,os,applications=[]):

  f = open("icinga_log.txt","a+")

  hosts = subprocess.check_output("curl -k -s -H 'Accept: application/json' -u 'admin:admin' -X GET 'http://192.168.6.65/icingaweb2/director/hosts'", shell=True)
  check_result = subprocess.check_output("curl -k -s -H 'Accept: application/json' -u 'admin:admin' -X GET 'http://192.168.6.65/icingaweb2/director/host?name=" + hostname + "'",
  shell=True)
  toutou = ""

  if ip in hosts:
    '''
    Host already exist in icinga
    Only existing templates will be added
    '''
    list_imports = re.search(r"\"imports\": \[[\n\r](?P<imports>[^]]+)",check_result)
    try:
      templates = list_imports.group('imports')
      temp_split = templates.split()
      temp_exist = ""
      for i in temp_split:
        temp_exist += i

      full_temp = temp_exist
      self.icinga_get_full_templates(os,f,applications)
      for i in applications:
        if i in templates:
          self.get_date_formatted()
          f.write(formatted + " - ICINGA - SUCCESS - Template " + i + " already exists on host " + ip + "\n")
        else:
          toutou = full_temp + "," + new_t_list
      if toutou != "":
        new_imports = subprocess.check_output("curl -k -s -H 'Accept: application/json' -u 'admin:admin' -X POST 'http://192.168.6.65/icingaweb2/director/host?name=" + hostname +"' -d
        '{\"imports\": [" + toutou + "]}'", shell=True)
        self.deploy_icinga(f)

    except AttributeError:
      self.get_date_formatted()
      f.write(formatted + " ICINGA - ERROR   - IP " + ip + " already exists\n")
    except:
      self.get_date_formatted()
      f.write(formatted + " - ICINGA - ERROR   - Something went wrong")

  elif hostname in hosts:
    self.get_date_formatted()
    f.write(formatted + " - ICINGA - ERROR   - Trying to recreate host '" + hostname + "'\n")

  else:
    '''
    Host does not exist in icinga
    He will be created
    Only existing templates will be added
    '''
    self.icinga_get_full_templates(os,f,applications)
    if new_t_list != "":
      create_host_cmd = subprocess.check_output("curl -k -s -H 'Accept: application/json' -u 'admin:admin' -X POST 'http://192.168.6.65/icingaweb2/director/host' -d '{\"object_name\": \""
      + hostname + "\",\"object_type\": \"object\",\"imports\": [" + new_t_list + "],\"address\": \"" + ip + "\"}'", shell=True)
      if "Traceback" in create_host_cmd:
        self.get_date_formatted()
        f.write(formatted + " - ICINGA - ERROR   - Host " + hostname + " with IP " + ip + " not created\n")
      else:
        self.deploy_icinga(f)
    else:
      self.get_date_formatted()
      f.write(formatted + " - ICINGA - ERROR   - An existing template is mandatory.\n")
```

Annexe C
Définition des paramètres du fichier CSV



Options

Global object type: Server

Separator: , "Wert","Wert2","..."

Headers: ✓

Consider default template: ✓

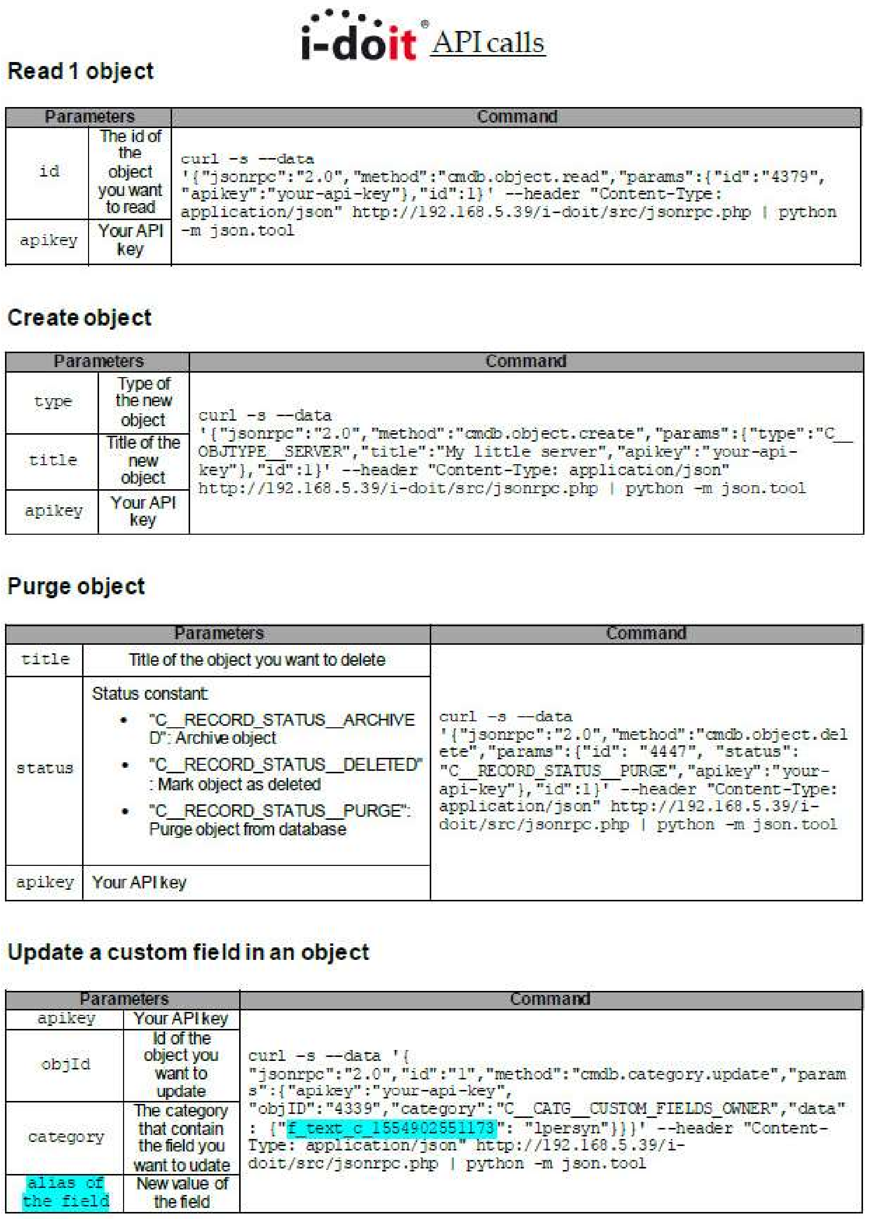
Prepare mapping

Adopt empty values: Yes [✓ No]

Multi-valued categories: [✓ Column] Row Comma-separated

Handling multi-valued categories:
[✓ Create category entries only if the category is empty (create if empty)]
Create category entries and keep existing ones (add)
Create category entries and replace existing ones (replace)

Annexe D
Aperçu du guide des appels à l’API d’i-doit



i-doit® API calls

Read 1 object

Parameters		Command
`id`	The id of the object you want to read	`curl -s --data '{"jsonrpc":"2.0","method":"cmdb.object.read","params":{"id":"4379", "apikey":"your-api-key"},"id":1}' --header "Content-Type: application/json" http://192.168.5.39/i-doit/src/jsonrpc.php | python -m json.tool`
`apikey`	Your API key	

Create object

Parameters		Command
`type`	Type of the new object	`curl -s --data '{"jsonrpc":"2.0","method":"cmdb.object.create","params":{"type":"C__OBJTYPE__SERVER","title":"My little server","apikey":"your-api-key"},"id":1}' --header "Content-Type: application/json" http://192.168.5.39/i-doit/src/jsonrpc.php | python -m json.tool`
`title`	Title of the new object	
`apikey`	Your API key	

Purge object

Parameters		Command
`title`	Title of the object you want to delete	`curl -s --data '{"jsonrpc":"2.0","method":"cmdb.object.delete","params":{"id": "4447", "status": "C__RECORD_STATUS__PURGE","apikey":"your-api-key"},"id":1}' --header "Content-Type: application/json" http://192.168.5.39/i-doit/src/jsonrpc.php | python -m json.tool`
`status`	Status constant: • "C__RECORD_STATUS__ARCHIVED": Archive object • "C__RECORD_STATUS__DELETED": Mark object as deleted • "C__RECORD_STATUS__PURGE": Purge object from database	
`apikey`	Your API key	

Update a custom field in an object

Parameters		Command
`apikey`	Your API key	`curl -s --data '{"jsonrpc":"2.0","id":"1","method":"cmdb.category.update","params":{"apikey":"your-api-key", "objID":"4339","category":"C__CATG__CUSTOM_FIELDS_OWNER","data": {"f_text_c_1554902551173": "1persyn"}}}' --header "Content-Type: application/json" http://192.168.5.39/i-doit/src/jsonrpc.php | python -m json.tool`
`objId`	Id of the object you want to update	
`category`	The category that contain the field you want to udate	
`alias of the field`	New value of the field	