

# Dolibarr dans RT

# Oeris

---

Dolibarr dans RT – Guide d'installation

---

Ce document a été rédigé par  **Oeris** pour  **Audiotronic**

Rédacteurs :

- Neil Orley
-

## Table des matières

1. Introduction.....	3
2. Préréquis / Préalable .....	4
a) Instance Dolibarr .....	4
b) Instance Request Tracker .....	4
3. Installation des développements Dolibarr .....	5
a) Activation du module « API/Web services (serveur SOAP) ».....	5
b) Génération d'une « dolibarrkey » .....	5
c) Importer les développements sur le serveur applicatif Dolibarr .....	5
d) Configuration du dictionnaire (optionnel) .....	6
e) Débogage.....	6
4. Installation des développements Request Tracker .....	7
a) Identifier les répertoires d'installation.....	7
b) Configuration de Request Tracker et installation des développements.....	8
i) Affichage du widget.....	8
ii) Configuration de l'authentification Dolibarr .....	8
iii) Configuration des filtres d'auto-completion .....	8
i) Configuration de l'auto-completion d'un seul champ personnalisé (optionnel) .....	9
ii) Configuration des champs personnalisés de retour.....	10
iii) Configuration du comportement de facturation .....	10
c) Débogage.....	10
5. Annexes .....	12
a) Scrip d'ajout de temps passé lors de la mise à jour du champ « Temps passé ».....	12
i) Configuration de la partie « Essentiel » .....	12
ii) Configuration de la partie « Conditions et actions définies par l'utilisateur ».....	12
(1) Condition personnalisée.....	12
(2) Programme de préparation d'action personnalisé .....	12
(3) Code d'action personnalisée (commit) .....	13
b) Scrip d'ajout de temps passé lors du changement de valeur du CP Facture .....	15
i) Configuration de la partie « Essentiel » .....	15
ii) Configuration de la partie « Conditions et actions définies par l'utilisateur ».....	15
(1) Condition personnalisée.....	15
(2) Programme de préparation d'action personnalisé .....	15
(3) Code d'action personnalisée (commit) .....	16

c)	Scrip d'ajout d'un forfait lors du changement de valeur du CP Facture .....	18
i)	Configuration de la partie « Essentiel » .....	18
ii)	Configuration de la partie « Conditions et actions définies par l'utilisateur ».....	18
(1)	Condition personnalisée.....	18
(2)	Programme de préparation d'action personnalisé .....	18
(3)	Code d'action personnalisée (commit) .....	19
d)	Scrip de changement d'état de la facture Dolibarr .....	20
i)	Configuration de la partie « Essentiel » .....	21
ii)	Configuration de la partie « Conditions et actions définies par l'utilisateur ».....	21
(1)	Condition personnalisée.....	21
(2)	Programme de préparation d'action personnalisé .....	21
(3)	Code d'action personnalisée (commit) .....	22
e)	Scrip d'ajout de demandeur lors de la complétion d'e-mail de Dolibarr .....	24
i)	Configuration de la partie « Essentiel » .....	24
ii)	Configuration de la partie « Conditions et actions définies par l'utilisateur ».....	24
(1)	Condition personnalisée.....	24
(2)	Programme de préparation d'action personnalisé .....	24
(3)	Code d'action personnalisée (commit) .....	25

## 1. Introduction

Ce guide présente l’installation et la configuration d’un module de création de facture et de gestion de contact dans Dolibarr, depuis RT.

## 2. Préréquis / Préalable

### a) Instance Dolibarr

- Version minimale : 5.0.0-b
- Accessible en http(s) depuis le serveur applicatif Request Tracker
- Le module « API/Web services (serveur SOAP) » activé et configuré
- Une « dolibarrkey » permettant d'authentifier le client SOAP
- Un accès SSH root au serveur applicatif Dolibarr

### b) Instance Request Tracker

- Version minimale : 4.4.1
- Un accès SSH root au serveur applicatif Dolibarr
- L'application GIT installé sur le serveur applicatif Request Tracker
- Packages PERL nécessaires :

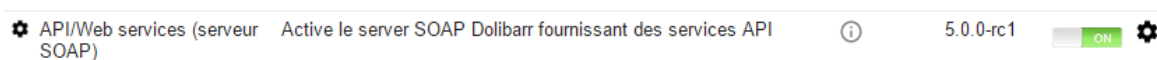
```
SOAP::Lite;  
MIME::Entity;  
HTTP::Cookies;  
HTTP::Message;  
HTTP::Response;  
HTTP::Request;
```

### 3. Installation des développements Dolibarr

La méthode ci-dessous décrit l'installation des développements Dolibarr sous forme d'évolution de l'interface SOAP. Les développements ne sont pas fournis sous forme de module Dolibarr.

#### a) Activation du module « API/Web services (serveur SOAP) ».

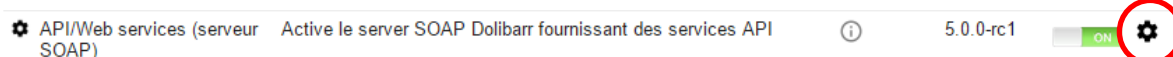
Activer le module depuis le menu « Configuration -> Modules » :



#### b) Génération d'une « dolibarrkey ».

Cette clé permet d'authentifier les clients du serveur SOAP, en plus de leurs identifiants / mot de passe.

Accéder au menu de paramétrage du module « API/Web services (serveur SOAP) ».



Générer la clé :

Paramètre	Valeur
Clé pour utiliser les Web Services (paramètre "dolibarrkey" dans les webservice)	6183aab932867d59653aeb30f46b2152



#### c) Importer les développements sur le serveur applicatif Dolibarr.

Il est recommandé d'utiliser un client SCP pour copier les fichiers nécessaires sur le serveur applicatif Dolibarr, ou de les télécharger directement depuis <https://bitbucket.org/oeris/ws-dolibarr/src>.

Les développements côté Dolibarr sont composés de 8 fichiers php :

- oeris\_get\_document.php
- oeris\_invoice.php
- oeris\_order.php
- oeris\_productorservice.php
- oeris\_searchcontact.php
- \* oeris\_thirdparty.php
- \* oeris\_user.php

Les développements sont à déposer dans le répertoire d'installation de Dolibarr de manière à obtenir l'arborescence suivante :

```
dolibarr
|   +--- htdocs
|   |
|   |   +--- webservices
|   |   |
|   |   |   +--- oeris_get_document.php
|   |   |   +--- oeris_order.php
|   |   |   +--- oeris_productorservice.php
|   |   |   +--- oeris_searchcontact.php
|   |   |   +--- oeris_user.php
|   |   |   +--- oeris_thirdparty.php
|   |   |   +--- oeris_invoice.php
```




















Note : Les fichiers peuvent être récupérés depuis <https://bitbucket.org/oeris/ws-dolibarr/src>

#### d) Configuration du dictionnaire (optionnel)

Pour éviter d'inscrire en banque une transaction qui n'est pas effectivement réalisée, il est possible de définir une condition de règlement spécifique dans Dolibarr.

La configuration se fait depuis le dictionnaire : « Configuration -> Dictionnaires -> Conditions de règlement »

Le code associé à cette condition de règlement doit être « MANUEL ». Ceci empêchera de classer la facture au statut « Payé » et de saisir le règlement en banque.

Code	Libellé*	Libellé sur les documents*	Nbre de jours	En fin de mois	Décalage	Ordre de tri	État
45DENDMONTH	45 fin de mois	Règlement à 45 jours fin de mois	45	En fin de mois			<input checked="" type="checkbox"/> ON  
RECEP	À réception	À réception de facture	0	Aucun	1	Toujours actif	
30D	30 jours	Règlement à 30 jours	30	Aucun	2		<input checked="" type="checkbox"/> ON  
30DENDMONTH	30 jours fin de mois	Règlement à 30 jours fin de mois	30	En fin de mois	3		<input checked="" type="checkbox"/> ON  
60D	60 jours	Règlement à 60 jours	60	Aucun	4		<input checked="" type="checkbox"/> ON  
60DENDMONTH	60 jours fin de mois	Règlement à 60 jours fin de mois	60	En fin de mois	5		<input checked="" type="checkbox"/> ON  
PT_ORDER	A commande	À réception de commande	0	Aucun	6		<input checked="" type="checkbox"/> ON  
PT_DELIVERY	A livraison	Règlement à la livraison	0	Aucun	7		<input checked="" type="checkbox"/> ON  
PT_5050	50/50	Règlement 50% d'avance, 50% à la livraison	0	Aucun	8		<input checked="" type="checkbox"/> ON  
MANUEL	Règlement saisi manuellement	Règlement saisi manuellement	0	Aucun	9		<input checked="" type="checkbox"/> ON  

#### e) Débogage

En cas d'erreurs, un message est ajouté dans les fichiers de logs du serveur http.

## 4. Installation des développements Request Tracker

### a) Identifier les répertoires d'installation.

Par convention les modifications apportées à Request Tracker sont réalisées dans le répertoire « local ».

Il est possible d'identifier le chemin d'accès à ce répertoire depuis le menu de configuration système de RT :

« Administration -> Outils -> Configuration Système »

Variables globales RT

Variable	Valeur
RT::BasePath	/usr/local
RT::BinPath	/usr/local/bin
RT::EtcPath	/usr/local/etc/rt44
RT::FontPath	/usr/local/share/rt44/fonts
RT::LexiconPath	/usr/local/share/rt44/po
RT::LocalEtcPath	/usr/local/www/rt44/etc
RT::LocalLexiconPath	/usr/local/www/rt44/po
RT::LocalLibPath	/usr/local/www/rt44/lib
RT::LocalPath	/usr/local/www/rt44
RT::LocalPluginPath	/usr/local/www/rt44/plugins
RT::LocalStaticPath	/usr/local/www/rt44/static
RT::MAJOR_VERSION	4
RT::MINOR_VERSION	4
RT::MasonComponentRoot	/usr/local/share/rt44/html
RT::MasonDataDir	/var/run/rt44/mason_data
RT::MasonLocalComponentRoot	/usr/local/www/rt44/html
RT::MasonSessionDir	/var/run/rt44/session_data
RT::PluginPath	/usr/local/plugins
RT::REVISION	1
RT::SbinPath	/usr/local/sbin
RT::StaticPath	/usr/local/share/rt44/static
RT::VERSION	4.4.1
RT::VarPath	/var/run/rt44

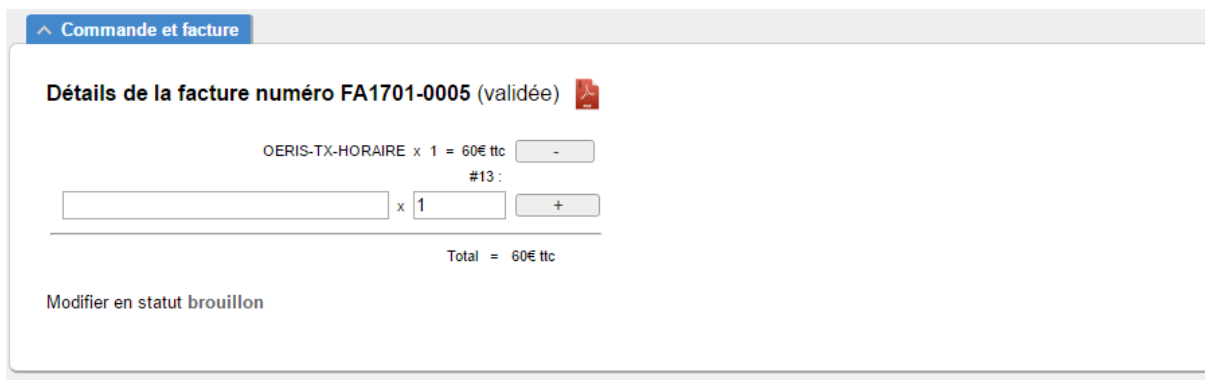
Dans certains cas il peut être nécessaire de modifier la configuration de Request Tracker pour déplacer l'emplacement du répertoire « local ».

La modification des répertoires utilisés par RT se fait depuis le fichier :  
 <RT Lib Path>/RT/Generated.pm



## b) Configuration de Request Tracker et installation des développements.

### i) Affichage du widget « Commande et facture ».



Le choix de l'affichage du widget se fait par file.

Pour définir sur quelles files l'afficher grâce à la variable `DolToRTQueues`

```
Set(@DolToRTQueues, qw/Audiotronic/);
```

### ii) Configuration de l'authentification Dolibarr.

```
# Dolibarr WS auth settings
Set($DolibarrURL, 'https://dolibarr.exemple.org');
Set($DolibarrKey, '1231445621458745428712176415687467');
Set($DolibarrLogin, 'John');
Set($DolibarrPassword, 'P@55w0rd');
```

### iii) Configuration des filtres d'auto-completion

Pour permettre la recherche des contacts Dolibarr depuis RT, il est nécessaire de définir une association entre les champs personnalisés RT et les champs retournés par les webservice Dolibarr.

Cette association est réalisée grâce à la variable `CFandFilter`

```
Set($CFandFilter, { 'Nom' => 'nom', 'Prenom' => 'prenom', 'E-mail' => 'email', 'Societe' =>
'societe', 'Telephone' => 'phone_pro', 'Telephone mobile' => 'phone_mobile', 'Telephone
perso' => 'phone_perso', } });
```

Les champs de recherche possibles dans Dolibarr depuis RT sont les suivants :

- nom : nom d'un contact Dolibarr
- prenom : prénom d'un contact Dolibarr
- email : e-mail d'un contact Dolibarr ou e-mail associé à un tiers Dolibarr
- societe : nom d'un tiers Dolibarr

- phone\_pro : téléphone associé à un tiers Dolibarr
- phone\_mobile : téléphone mobile associé à un contact Dolibarr
- phone\_perso : téléphone associé à un contact Dolibarr

En se basant sur la configuration ci-dessous, le fonctionnement de la recherche est le suivant :

Configuration : 'Nom' => 'nom', 'Prenom' => 'prenom', 'E-mail' => 'email', 'Societe' => 'societe', 'Telephone' => 'phone\_pro', 'Telephone mobile' => 'phone\_mobile', 'Telephone perso' => 'phone\_perso'.

Si recherche d'un contact à partir des champs personnalisés « Societe » ou « Telephone » :

- Le champ Societe est renseigné avec le nom du Tiers Dolibarr
- L'Email est renseigné avec l'email associée Tiers Dolibarr
- Le Téléphone est renseigné avec le phone\_pro du Tiers Dolibarr
- Le nom est renseigné avec le nom du Tiers Dolibarr
- Le prénom est laissé vide

Si recherche d'un contact à partir du champ « Nom », « Prenom », « E-mail », « Telephone perso » ou « Telephone mobile » :

- Le champ Societe est renseigné avec le nom du Tiers Dolibarr
- L'Email est renseigné avec l'email du contact Dolibarr
- Le Téléphone pro est renseigné avec le phone pro du Tiers Dolibarr
- Le Téléphone perso est renseigné avec le phone perso du contact Dolibarr
- Le Téléphone mobile est renseigné avec le phone mobile du contact Dolibarr
- Le nom est renseigné avec le nom du contact Dolibarr
- Le prénom est renseigné avec le prénom du contact Dolibarr

### i) Configuration de l'auto-complétion d'un seul champ personnalisé (optionnel)

Lors de la recherche de contact depuis les champs d'auto-complétion définis précédemment, il est possible de faire en sorte que seul le champ sur lequel la recherche a été réalisée soit auto-complété.

Pour cela il est nécessaire de définir la variable `SingleAutocomplete`

```
Set(@SingleAutocomplete, qw/Societe/);
```

Dans ce cas, lors de la recherche d'un nom de tiers Dolibarr, seul le champ Societe sera rempli avec la valeur retournée par Dolibarr. Les autres champs resteront non renseignés.

## ii) Configuration des autres champs personnalisés

Ces champs personnalisés sont utilisés pour stocker des informations contenues dans Dolibarr telles que la référence de la proposition commerciale, la référence de la facture ou le montant TTC de la facture.

Cette association est réalisée grâce aux variables ci-dessous :

```
Set($CFRefFacture, 'Facture'); # Peut être renseigné manuellement
Set($CFRefCommand, 'Commande'); # Doit être renseigné manuellement
Set($CFTotalTTC, 'Total TTC'); # Est renseigné automatiquement
```

## iii) Configuration du comportement de facturation

Pour chaque file, il est possible d'intégrer l'ajout d'une ligne de facturation automatiquement pour chaque référence de facture contenue dans le champ personnalisé précédemment défini.

Ce type de facturation peut prendre 3 formes :

- Au temps passé : la valeur du champ RT « Temps passé » est alors prise en compte
- Au forfait : un forfait est ajouté
- Aucune facturation particulière

Pour cela il est nécessaire de définir un « Produit / Service » dans Dolibarr décrivant a minima le montant associé à ce « Produit / Service » et de déclarer le comportement de chacune des files dans au moyen de la variable `BillingTypePerQueue`

```
Set($BillingTypePerQueue, { { 'Queue1' => { 'forfait' => 'OERIS-FORFAIT' } },
                           { 'Queue2' => { 'temps' => 'OERIS-TX-HORAIRE' } } });
```

Dans l'exemple de configuration ci-dessus, un produit 'OERIS-FORFAIT' et un produit 'OERIS-TX-HORAIRE' ont été définis dans Dolibarr.

L'ajout de la ligne de facturation est alors déclenché au moyen de scrips RT (cf : annexes), sur des conditions particulières paramétrables.

## c) Débogage.

Pour simplifier le débogage, il est recommandé de configurer RT de cette manière :

```
Set($LogToFile, 'debug');
Set($DevelMode, '1');
```

Lorsque la variable `LogToFile` vaut `'debug'`, à chaque appel d'un webservice un fichier est généré dans `/tmp`. Ces fichiers contiennent l'ensemble des échanges SOAP au format XML. De plus les logs RT sont alimentés avec les commentaires intégrés dans le code ou les scripts.

## 5. Annexes

Ces annexes décrivent chaque fonctionnalité implémentée par scrip. A noter qu'il peut être nécessaire d'implémenter plusieurs fois le même scrip pour permettre le fonctionnement désiré et/ou de définir des contions personnalisées.

### a) Scrip d'ajout de temps passé lors de la mise à jour du champ « Temps passé »

#### i) Configuration de la partie « Essentiel »

Note : ce scrip doit être configuré pour s'exécuter à l' »Etape » « Batch »

Description:	Dolibarr Temps passé lors d'une mise a jour du temps passé
Condition:	Lors d'un changement de temps passé ▼
Action:	Défini par l'utilisateur ▼
Modèle:	Blank ▼
S'applique à:	Global
	<input checked="" type="checkbox"/> Activé (Décocher cette case pour désactiver ce scrip)

#### ii) Configuration de la partie « Conditions et actions définies par l'utilisateur »

#### (1)Condition personnalisée

```
return 1;
```

#### (2)Programme de préparation d'action personnalisé

```
#---- Récupère le type de facturation par file: par défaut au forfait
my $BillingTypePerQueue = RT->Config->Get('BillingTypePerQueue');

my $CurrentQueue = $self->TicketObj->QueueObj->Name;
my @DolToRTQueues = RT->Config->Get('DolToRTQueues');

if( defined($CurrentQueue) && scalar(grep(/$CurrentQueue/, @DolToRTQueues)) ) {
    #---- Ajoute le nombre d'heures passées a la facture
    unless(defined($BillingTypePerQueue->{$CurrentQueue}->{'temps'})) {
        $RT::Logger->debug( "[Scrip Facturation#1] La file ".$CurrentQueue." est au forfait" );
        return 0;
    }
} else {
```

```
$RT::Logger->debug( "[Scrip Facturation#1] Ce scrip ne s'applique pas a la file
".$CurrentQueue." );
return 0;
}
```

### (3) Code d'action personnalisée (commit)

```
use SOAP::Lite;
use MIME::Entity;
use HTTP::Cookies;
use HTTP::Message;
use HTTP::Response;
use HTTP::Request;
use Data::Dumper;

#---- Récupère le type de facturation par file: par défaut au forfait
my $BillingTypePerQueue = RT->Config->Get('BillingTypePerQueue');

my $CurrentQueue = $self->TicketObj->QueueObj->Name;

#---- Récupère le numéro de facture
my $ref_facture = $self->TicketObj->FirstCustomFieldValue(RT->Config->Get('CRefFacture'));
if (!defined($ref_facture)){
    $RT::Logger->error( "[Scrip Facturation#1] Aucune référence de facture renseignée");
    return 0;
}

#---- Récupère le temps travaillé
my $HourWorked = sprintf("%.2f", ($self->TicketObj->TimeWorked()/60));
my $Delta= sprintf("%.2f", ( ($self->TicketObj->TimeWorked() - $self->TransactionObj->OldValue)/60 ));
if ($HourWorked==0){
    $RT::Logger->error( "[Scrip Facturation#1] Temps travaillé vaut 0" );
    return 0;
}
$RT::Logger->debug( "[Scrip Facturation#1] Temps travaillé = ".$HourWorked.", Delta = ".$Delta);

my $soap = SOAP::Lite->proxy(RT->Config->Get( 'DolibarrURL '
).'/webservices/oeris_invoice.php', timeout => 10);
$soap->serializer->namespaces->{"ns"} = "xmlns:ns";
$soap->autotype( 0 );
$soap->soapversion( '1.1' );
$soap->ns( 'http://www.dolibarr.org/ns/', 'ns' );
$soap->envprefix( 'soapenv' );

my $header = SOAP::Header->type( 'xml' => '' );

my $auth = SOAP::Data->type("ns:authentication")->name('authentication' => \SOAP::Data->value(
    >name('dolibarrkey')->value(RT->Config->Get( 'DolibarrKey' )),          SOAP::Data-
    >name('sourceapplication')->value('Request Tracker'),                  SOAP::Data-
    >name('login')->value(RT->Config->Get( 'DolibarrLogin' )),              SOAP::Data-
```

```

>name('password')->value(RT->Config->Get( 'DolibarrPassword' )),
>name('entity')->value('')
);

my $invoice = SOAP::Data->type("ns:invoice")->name('invoice' => \SOAP::Data->value(
    >type("xsd:string")->name('ref')->value($ref_facture),
    >type("ns:LinesArray2")->name('lines' => \SOAP::Data->value(
        SOAP::Data->type("ns:line")->name('line' => \SOAP::Data->value(
            SOAP::Data->type("xsd:double")->name('qty')->value($HourWorked),
            SOAP::Data->type("xsd:string")->name('desc')->value('#'.$self->TicketObj->Id.' : '.$self->TicketObj->Subject()),
            SOAP::Data->type("xsd:double")->name('total_qty')->value($HourWorked),
            SOAP::Data->type("xsd:string")->name('product_ref')->value($BillingTypePerQueue->{$CurrentQueue}->{'temps'})
        )
    )
);

my $som = $soap->call('updateInvoice',
    $header,
    $auth,
    $invoice
);

#---- Erreur dans le retour du WS
if ( $som->fault ) {
    $RT::Logger->error ( "[Scrip Facturation#1] Error when calling webservice '".RT->Config->Get( 'DolibarrURL' )."/webservices/oeris_invoice.php#updateInvoice' :". $som->fault->{'faultstring'} );
} elseif ($som->result->{'result_code'} eq 'OK') {
    $RT::Logger->debug( "[Scrip Facturation#1] Mise a jour de la facture ".$ref_facture.". Temps travaillé = ".$HourWorked );
} else {
    $RT::Logger->error ( "[Scrip Facturation#1] Error when calling webservice '".RT->Config->Get( 'DolibarrURL' )."/webservices/oeris_invoice.php#updateInvoice' :". $som->result->{'result_label'} );
}

1;

```

## b) Scrip d'ajout de temps passé lors du changement de valeur du CP Facture

### i) Configuration de la partie « Essentiel »

Note : ce scrip doit être configuré pour s'exécuter à l' »Etape » « Batch »

Description:	Dolibarr Temps passé lors du changement de valeur du CP Facture
Condition:	Défini par l'utilisateur ▼
Action:	Défini par l'utilisateur ▼
Modèle:	Blank ▼
S'applique à:	Global
	<input checked="" type="checkbox"/> Activé (Décocher cette case pour désactiver ce scrip)

### ii) Configuration de la partie « Conditions et actions définies par l'utilisateur »

#### (1) Condition personnalisée

```
my $cfname = RT->Config->Get('CfRefFacture');
my $cf = RT::CustomField->new(RT->SystemUser);
$cf->LoadByName(Name => $cfname);
my $cfid = $cf->Id();

# If create or change to custom field
unless ( ( $self->TransactionObj->Type eq "CustomField" && $self->TransactionObj->Field == $cfid ) || $self->TransactionObj->Type eq "Create" ) {
    return 0;
}

return 1;
```

#### (2) Programme de préparation d'action personnalisé

```
#---- Récupère le type de facturation par file: par défaut au forfait
my $BillingTypePerQueue = RT->Config->Get('BillingTypePerQueue');

my $CurrentQueue = $self->TicketObj->QueueObj->Name;
my @DolToRTQueues = RT->Config->Get('DolToRTQueues');

if( defined($CurrentQueue) && scalar(grep(/$CurrentQueue/, @DolToRTQueues)) ) {
    #---- Ajoute le nombre d'heures passées a la facture
    unless(defined($BillingTypePerQueue->{$CurrentQueue}->{'temps'})) {
        $RT::Logger->debug( "[Scrip Facturation#1c] La file ".$CurrentQueue." est au forfait" );
    };
    return 0;
}
} else {
```



```
$RT::Logger->debug( "[Scrip Facturation#1c] Ce scrip ne s'applique pas a la file
".$CurrentQueue." ");
return 0;
}
```

### (3) Code d'action personnalisée (commit)

```
use SOAP::Lite;
use MIME::Entity;
use HTTP::Cookies;
use HTTP::Message;
use HTTP::Response;
use HTTP::Request;
use Data::Dumper;

#---- Récupère le type de facturation par file: par défaut au forfait
my $BillingTypePerQueue = RT->Config->Get('BillingTypePerQueue');

my $CurrentQueue = $self->TicketObj->QueueObj->Name;

#---- Récupère le numéro de facture
my $ref_facture = $self->TicketObj->FirstCustomFieldValue(RT->Config->Get('CRefFacture'));
if (!defined($ref_facture)){
    $RT::Logger->error( "[Scrip Facturation#1c] Aucune référence de facture renseignée");
    return 0;
}

#---- Récupère le temps travaillé
my $HourWorked = sprintf("%.2f", ($self->TicketObj->TimeWorked()/60));
my $Delta= sprintf("%.2f", ( ($self->TicketObj->TimeWorked() - $self->TransactionObj->OldValue)/60 ));
if ($HourWorked==0){
    $RT::Logger->error( "[Scrip Facturation#1c] Temps travaillé vaut 0" );
    return 0;
}
$RT::Logger->debug( "[Scrip Facturation#1c] Temps travaillé = ".$HourWorked.", Delta = ".$Delta);

my $soap = SOAP::Lite->proxy(RT->Config->Get( 'DolibarrURL '
).'/webservices/oeris_invoice.php', timeout => 10);
$soap->serializer->namespaces->{"ns"} = "xmlns:ns";
$soap->autotype( 0 );
$soap->soapversion( '1.1' );
$soap->ns( 'http://www.dolibarr.org/ns/', 'ns' );
$soap->envprefix( 'soapenv' );

my $header = SOAP::Header->type( 'xml' => '' );

my $auth = SOAP::Data->type("ns:authentication")->name('authentication' => \SOAP::Data-
>value(
    >name('dolibarrkey')->value(RT->Config->Get( 'DolibarrKey' )),          SOAP::Data-
    >name('sourceapplication')->value('Request Tracker'),                  SOAP::Data-
    >name('login')->value(RT->Config->Get( 'DolibarrLogin' )),              SOAP::Data-
```

```

>name('password')->value(RT->Config->Get( 'DolibarrPassword' )),
>name('entity')->value('')
    );

my $invoice = SOAP::Data->type("ns:invoice")->name('invoice' => \SOAP::Data->value(
    >type("xsd:string")->name('ref')->value($ref_facture),
    >type("ns:LinesArray2")->name('lines' => \SOAP::Data->value(
        SOAP::Data->type("ns:line")->name('line' => \SOAP::Data->value(
            SOAP::Data->type("xsd:double")->name('qty')->value($HourWorked),
            SOAP::Data->type("xsd:string")->name('desc')->value('#'.$self->TicketObj->Id.' : '.$self->TicketObj->Subject()),
            SOAP::Data->type("xsd:double")->name('total_qty')->value($HourWorked),
            SOAP::Data->type("xsd:string")->name('product_ref')->value($BillingTypePerQueue->{$CurrentQueue}->{'temps'})
        )
    )
)
);

my $som = $soap->call('updateInvoice',
    $header,
    $auth,
    $invoice
);

#---- Erreur dans le retour du WS
if ( $som->fault ) {
    $RT::Logger->error ( "[Scrip Facturation#1c] Error when calling webservice '".RT->Config->Get( 'DolibarrURL' )."/webservices/oeris_invoice.php#updateInvoice' :". $som->fault->{'faultstring'} );
} elseif ($som->result->{'result_code'} eq 'OK') {
    $RT::Logger->debug( "[Scrip Facturation#1c] Mise a jour de la facture ".$ref_facture.". Temps travaillé = ".$HourWorked );
} else {
    $RT::Logger->error ( "[Scrip Facturation#1c] Error when calling webservice '".RT->Config->Get( 'DolibarrURL' )."/webservices/oeris_invoice.php#updateInvoice' :". $som->result->{'result_label'} );
}

1;

```

## c) Scrip d'ajout d'un forfait lors du changement de valeur du CP Facture

### i) Configuration de la partie « Essentiel »

Note : ce scrip doit être configuré pour s'exécuter à l' »Etape » « Batch »

Description:	Dolibarr ajout ligne forfait en Facturation lors du changement de valeur du CP
Condition:	Défini par l'utilisateur ▼
Action:	Défini par l'utilisateur ▼
Modèle:	Blank ▼
S'applique à:	Global
	<input checked="" type="checkbox"/> Activé (Décocher cette case pour désactiver ce scrip)

### ii) Configuration de la partie « Conditions et actions définies par l'utilisateur »

#### (1)Condition personnalisée

```
my $cfname = RT->Config->Get('CFRefFacture');
my $cf = RT::CustomField->new(RT->SystemUser);
$cf->LoadByName(Name => $cfname);
my $cfid = $cf->Id();

# If create or change to custom field
unless ( ( $self->TransactionObj->Type eq "CustomField" && $self->TransactionObj->Field == $cfid ) || $self->TransactionObj->Type eq "Create" ) {
    return 0;
}

return 1;
```

#### (2)Programme de préparation d'action personnalisé

```
my $CurrentQueue = $self->TicketObj->QueueObj->Name;
my @Do1ToRTQueues = RT->Config->Get('Do1ToRTQueues');

if( defined($CurrentQueue) && scalar(grep(/$CurrentQueue/, @Do1ToRTQueues)) ) {

    #--- Récupère la référence de la facture
    my $ref_invoice = $self->TicketObj->FirstCustomFieldValue(RT->Config->Get('CFRefFacture'));

    if (defined($ref_invoice)) {
        return 1;
    } else {
        $RT::Logger->error ( "[Scrip Facturation#2c] Aucune référence de facture renseignée");
        return 0;
    }
}
```

```

    }
} else {
    $RT::Logger->error ( "[Scrip Facturation#2c] Ce scrip ne s'applique pas a la file
    ".$CurrentQueue);
    return 0;
}

```

### (3)Code d'action personnalisée (commit)

```

use SOAP::Lite;
use MIME::Entity;
use HTTP::Cookies;
use HTTP::Message;
use HTTP::Response;
use HTTP::Request;
use Data::Dumper;

#---- Ajoute le forfait de facturation si besoin

#---- Récupère le type de facturation par file: par défaut au forfait
my $BillingTypePerQueue = RT->Config->Get('BillingTypePerQueue');

#---- Récupère la référence de la facture
my $ref_invoice = $self->TicketObj->FirstCustomFieldValue(RT->Config->Get('CRefFacture'));

#---- Récupère le nom de la file
my $CurrentQueue = $self->TicketObj->QueueObj->Name;

my $soap = undef;
my $header = undef;
my $auth = undef;
my $invoice = undef;
my $som = undef;

#---- Ajoute le nombre d'heures passées a la facture
if(defined($BillingTypePerQueue->{$CurrentQueue}->{'forfait'})) {
    $RT::Logger->debug( "[Scrip Facturation#2c] Le forfait ".$BillingTypePerQueue-
    >{$CurrentQueue}->{'forfait'}." a la file ".$CurrentQueue);

    $soap = SOAP::Lite->proxy(RT->Config->Get( 'DolibarrURL'
    ).'/webservises/oeris_invoice.php', timeout => 10);
    $soap->serializer->namespaces->{"ns"} = "xmlns:ns";
    $soap->autotype( 0 );
    $soap->soapversion( '1.1' );
    $soap->ns( 'http://www.dolibarr.org/ns/', 'ns' );
    $soap->envprefix( 'soapenv' );

    $header = SOAP::Header->type( 'xml' => '' );

    $auth = SOAP::Data->type("ns:authentication")->name('authentication' => \SOAP::Data-
    >value(
    >name('dolibarrkey')->value(RT->Config->Get( 'DolibarrKey' )),
    >name('sourceapplication')->value('Request Tracker'),
    >name('login')->value(RT->Config->Get( 'DolibarrLogin' )),
    >name('password')->value(RT->Config->Get( 'DolibarrPassword' )),
    SOAP::Data-
    SOAP::Data-
    SOAP::Data-
    SOAP::Data-

```

```

>name('entity')->value('')
);

$invoice = SOAP::Data->type("ns:invoice")->name('invoice' => \SOAP::Data->value(
    SOAP::Data->type("xsd:string")->name('ref')->value($ref_invoice),
    SOAP::Data->type("ns:LinesArray2")->name('lines' => \SOAP::Data->value(
        SOAP::Data->type("ns:line")->name('line' => \SOAP::Data->value(
            SOAP::Data->type("xsd:double")->name('qty')->value(1),
            SOAP::Data->type("xsd:double")->name('total_qty')->value(1),
            SOAP::Data->type("xsd:string")->name('desc')->value('#'.$self->TicketObj->Id.' : '.$self->TicketObj->Subject()),
            SOAP::Data->type("xsd:int")->name('product_id')->value(),
            SOAP::Data->type("xsd:string")->name('product_ref')->value($BillingTypePerQueue->{$CurrentQueue}->{'forfait'}))
        )
    )
);

$som = $soap->call('updateInvoice',
    $header,
    $auth,
    $invoice
);

#---- Erreur dans le retour du WS
if ( $som->fault ) {
    $RT::Logger->error ( "[Scrip Facturation#2c] Error when calling webservice '".RT-
>Config->Get( 'DolibarrURL' )."/webservices/oeris_invoice.php#updateInvoice' : " . $som-
>fault->{'faultstring'} );
    return 0;
} elseif ($som->result->{'result_code'} eq 'OK') {
    $RT::Logger->debug( "[Scrip Facturation#2c] Mise a jour de la facture ".$ref_invoice.".
Application du forfait ".$BillingTypePerQueue->{$CurrentQueue}->{'forfait'} );
} else {
    $RT::Logger->error ( "[Scrip Facturation#2c] Error when calling webservice '".RT-
>Config->Get( 'DolibarrURL' )."/webservices/oeris_invoice.php#updateInvoice' : " . $som-
>result->{'result_label'} );
    return 0;
}
} else {
    $RT::Logger->debug( "[Scrip Facturation#2c] Aucun forfait défini pour la file
".$CurrentQueue);
}

return 1;

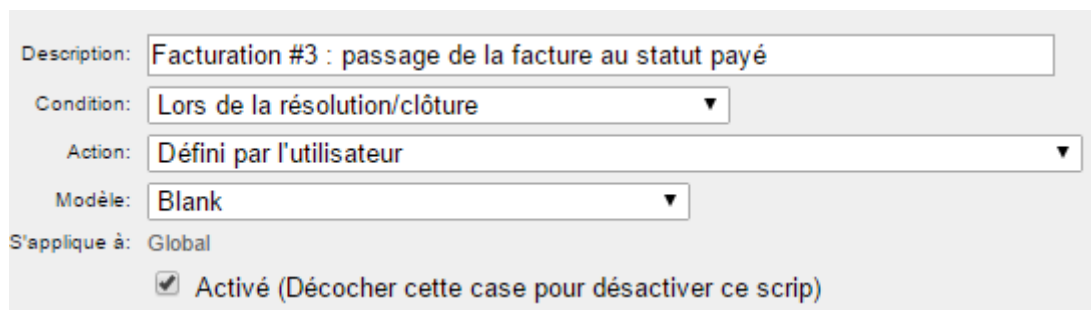
```

#### d) Scrip de changement d'état de la facture Dolibarr

Ce scrip permet de classer la facture au statut « Payé » dans Dolibarr et d'intégrer le montant en banque si la condition de règlement le permet (cf : 3.d)

### i) Configuration de la partie « Essentiel »

Note : ce scrip doit être configuré pour s'exécuter à l' « Etape » « Normal »



Description: Facturation #3 : passage de la facture au statut payé

Condition: Lors de la résolution/clôture ▼

Action: Défini par l'utilisateur ▼

Modèle: Blank ▼

S'applique à: Global

☒ Activé (Décocher cette case pour désactiver ce scrip)

### ii) Configuration de la partie « Conditions et actions définies par l'utilisateur »

#### (1) Condition personnalisée

Note : il est nécessaire de définir une condition personnalisée, notamment lors de l'utilisation du module [RT::Lifecycle](#)

```
return 1;
```

#### (2) Programme de préparation d'action personnalisé

```
my $CurrentQueue = $self->TicketObj->QueueObj->Name;
my @DolToRTQueues = RT->Config->Get('DolToRTQueues');

if( defined($CurrentQueue) && scalar(grep(/$CurrentQueue/, @DolToRTQueues)) ) {

    #--- Récupère la référence de la facture
    my $ref_invoice = $self->TicketObj->FirstCustomFieldValue(RT->Config->Get('CFRefFacture'));

    if (defined($ref_invoice)) {
        return 1;
    } else {
        $RT::Logger->error ( "[Scrip Facturation#3] Aucune référence de facture renseignée");
        return 0;
    }
} else {
    $RT::Logger->error ( "[Scrip Facturation#3] Ce scrip ne s'applique pas a la file ".$CurrentQueue);
    return 0;
}
```

### (3) Code d'action personnalisée (commit)

```

use SOAP::Lite;
use MIME::Entity;
use HTTP::Cookies;
use HTTP::Message;
use HTTP::Response;
use HTTP::Request;
use Data::Dumper;

#---- Ajoute le forfait de facturation si besoin

#---- Récupère le type de facturation par file: par défaut au forfait
my $BillingTypePerQueue = RT->Config->Get('BillingTypePerQueue');

#---- Récupère la référence de la facture
my $ref_invoice = $self->TicketObj->FirstCustomFieldValue(RT->Config->Get('CRefFacture'));

#---- Récupère le nom de la file
my $CurrentQueue = $self->TicketObj->QueueObj->Name;

my $soap = undef;
my $header = undef;
my $auth = undef;
my $invoice = undef;
my $som = undef;

#---- Passe la facture au statut payé

$soap = SOAP::Lite->proxy(RT->Config->Get( 'DolibarrURL' ).'/dolinrt/oeris_invoice.php',
timeout => 10);
$soap->serializer->namespaces->{"ns"} = "xmlns:ns";
$soap->autotype( 0 );
$soap->soapversion( '1.1' );
$soap->ns( 'http://www.dolibarr.org/ns/', 'ns' );
$soap->envprefix( 'soapenv' );

$header = SOAP::Header->type( 'xml' => '' );

$auth = SOAP::Data->type("ns:authentication")->name('authentication' => \SOAP::Data->value(
SOAP::Data-
>name('dolibarrkey')->value(RT->Config->Get( 'DolibarrKey' )),
SOAP::Data-
>name('sourceapplication')->value('Request Tracker'),
SOAP::Data-
>name('login')->value(RT->Config->Get( 'DolibarrLogin' )),
SOAP::Data-
>name('password')->value(RT->Config->Get( 'DolibarrPassword' )),
SOAP::Data-
>name('entity')->value('')
)
);

$invoice = SOAP::Data->type("ns:invoice")->name('invoice' => \SOAP::Data->value(
SOAP::Data-
>type("xsd:string")->name('ref')->value($ref_invoice),
SOAP::Data-
>type("xsd:int")->name('status')->value(2)
)
);

```

```
#---- Classe la facture au statut payé
$som = $soap->call('updateInvoice',
                  $header,
                  $auth,
                  $invoice
);

#---- Erreur dans le retour du WS
if ( $som->fault ) {
    $RT::Logger->error( "[Scrip Facturation#3] Error when calling webservice '".RT->Config-
>Get( 'DolibarrURL' )."/dolinrt/oeris_invoice.php#updateInvoice' :". $som->fault-
>{faultstring} );
    return 0;
} elseif ($som->result->{'result_code'} eq 'OK') {
    $RT::Logger->debug( "[Scrip Facturation#3] Mise a jour de la facture ".$ref_invoice.".
Statut = payé" );
} else {
    $RT::Logger->error( "[Scrip Facturation#3] Error when calling webservice '".RT->Config-
>Get( 'DolibarrURL' )."/dolinrt/oeris_invoice.php#updateInvoice' :". $som->result-
>{'result_label'} );
    return 0;
}

return 1;
```



## e) Scrip d'ajout de demandeur lors de la complétion d'e-mail de Dolibarr

Ce scrip permet d'ajouter un contact Dolibarr en demandeur d'un ticket RT

### i) Configuration de la partie « Essentiel »

Note : ce scrip doit être configuré pour s'exécuter à l' « Etape » « Normal »

Description:	On Update : ajoute le demandeur d'après Dolibarr
Condition:	Défini par l'utilisateur ▼
Action:	Défini par l'utilisateur ▼
Modèle:	Blank ▼
S'applique à:	Global
<input checked="" type="checkbox"/> <b>Activé (Décocher cette case pour désactiver ce scrip)</b>	

### ii) Configuration de la partie « Conditions et actions définies par l'utilisateur »

#### (1) Condition personnalisée

```
#---- Récupère la valeur de l'Email renseigné lors de la création du ticket
my $CFandFilter = RT->Config->Get('CFandFilter');
my %reverseCFandFilter = reverse %$CFandFilter;
my $CFEmail = $reverseCFandFilter{'email'};

my $CF = RT::CustomField->new(RT->SystemUser);
$CF->LoadByName(Name => $CFEmail);
my $CFEmailId = $CF->Id();

#---- Sur creation ou changement de valeur dans le CF Email
if( ($self->TransactionObj->Type eq "CustomField" && $self->TransactionObj->Field ==
$CFEmailId) || $self->TransactionObj->Type eq "Create" ) {
    $RT::Logger->debug( "[Scrip On Update: ajoute le demandeur d'après Dolibarr] Création ou
modification de la valeur du CP = ".$CFEmail);
    return 1;
} else {
    $RT::Logger->debug( "[Scrip On Update: ajoute le demandeur d'après Dolibarr] Aucune
action a réaliser = return 0");
    return 0;
}
```

#### (2) Programme de préparation d'action personnalisé

```
#---- Récupère la valeur de l'Email renseigné lors de la création du ticket
my $CFandFilter = RT->Config->Get('CFandFilter');
my %reverseCFandFilter = reverse %$CFandFilter;
my $CFEmail = $reverseCFandFilter{'email'};
my $email = $self->TicketObj->FirstCustomFieldValue($CFEmail);
```

```
$RT::Logger->debug( "[Scrip On Update: ajoute le demandeur d'après Dolibarr] Nombre de
demandeurs présents = ".scalar(split(/, /,$self->TicketObj->RequestorAddresses));

#---- Valide le format de l'adresse mail saisie dans le CP
if ($email =~ /^[a-z0-9.]+\@[a-z0-9.-]+\$/){
    $RT::Logger->debug( "[Scrip On Update: ajoute le demandeur d'après Dolibarr] Email
renseigné = ".$email);

    #---- Vérifie que l'email n'est pas déjà dans le champ demandeur
    if($self->TicketObj->RequestorAddresses =~ /$email/) {
        $RT::Logger->debug( "[Scrip On Update: ajoute le demandeur d'après Dolibarr] ".$email."
est déjà demandeur du ticket");
        return 0;
    } else {
        return 1;
    }
} else {
    $RT::Logger->debug( "[Scrip On Update: ajoute le demandeur d'après Dolibarr] ".$email." :
email invalide");
    return 0;
}
}
```

### (3) Code d'action personnalisée (commit)

```
#---- Récupère la valeur de l'Email renseigné lors de la création du ticket
my $CFandFilter = RT->Config->Get('CFandFilter');
my %reveseCFandFilter = reverse %$CFandFilter;
my $CFEmail = $reveseCFandFilter{'email'};
my $email = $self->TicketObj->FirstCustomFieldValue($CFEmail);

#---- Ajoute cet email en demandeur
my $UserObj = RT::User->new($RT::SystemUser);
$UserObj->LoadByEmail($email);
$self->TicketObj->Requestors->AddMember($UserObj->Id);

$RT::Logger->debug( "[Scrip On Update: ajoute le demandeur d'après Dolibarr] Nombre de
demandeurs présents = ".scalar(split(/, /,$self->TicketObj->RequestorAddresses)). " :
".$self->TicketObj->RequestorAddresses);

1;
```