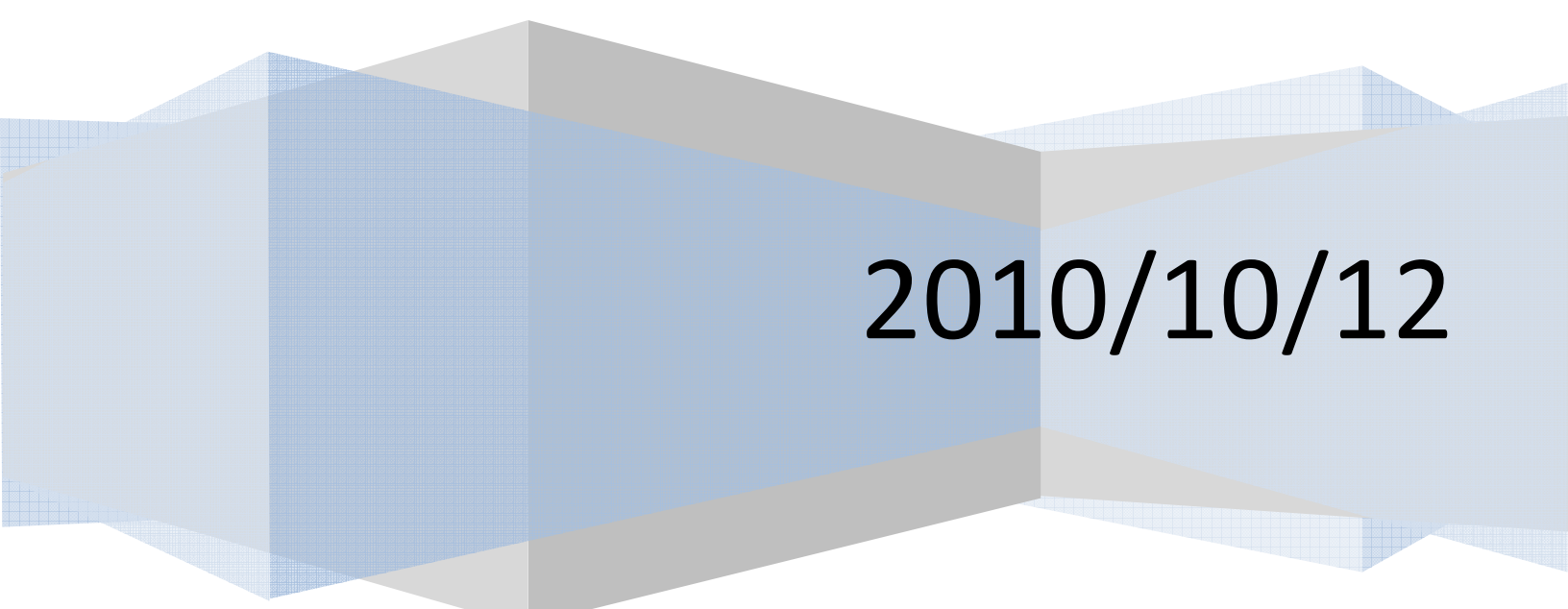


Paterva (Pty) Ltd

Integration options with Maltego v3

RT



2010/10/12

Integration options with Maltego v3

Table of Contents

| | |
|--|----|
| Introduction | 3 |
| TDS server | 4 |
| Seed management | 4 |
| Transform settings | 5 |
| Transforms | 6 |
| Discovering from the Maltego GUI | 7 |
| The transform in code..... | 9 |
| MaltegoEntity class:..... | 10 |
| MaltegoTransform class:..... | 10 |
| MaltegoMsg class:..... | 10 |
| SQLTAS | 13 |
| Connecting to the server with a browser: | 13 |
| Procedure..... | 17 |
| Mapping language | 18 |
| Reading Entities from the Maltego client (SQL query) | 18 |
| Writing entities to the Maltego client | 18 |
| Running a transform via the SQLTAS on the client | 19 |
| Running the discovered transform | 22 |

Integration options with Maltego v3

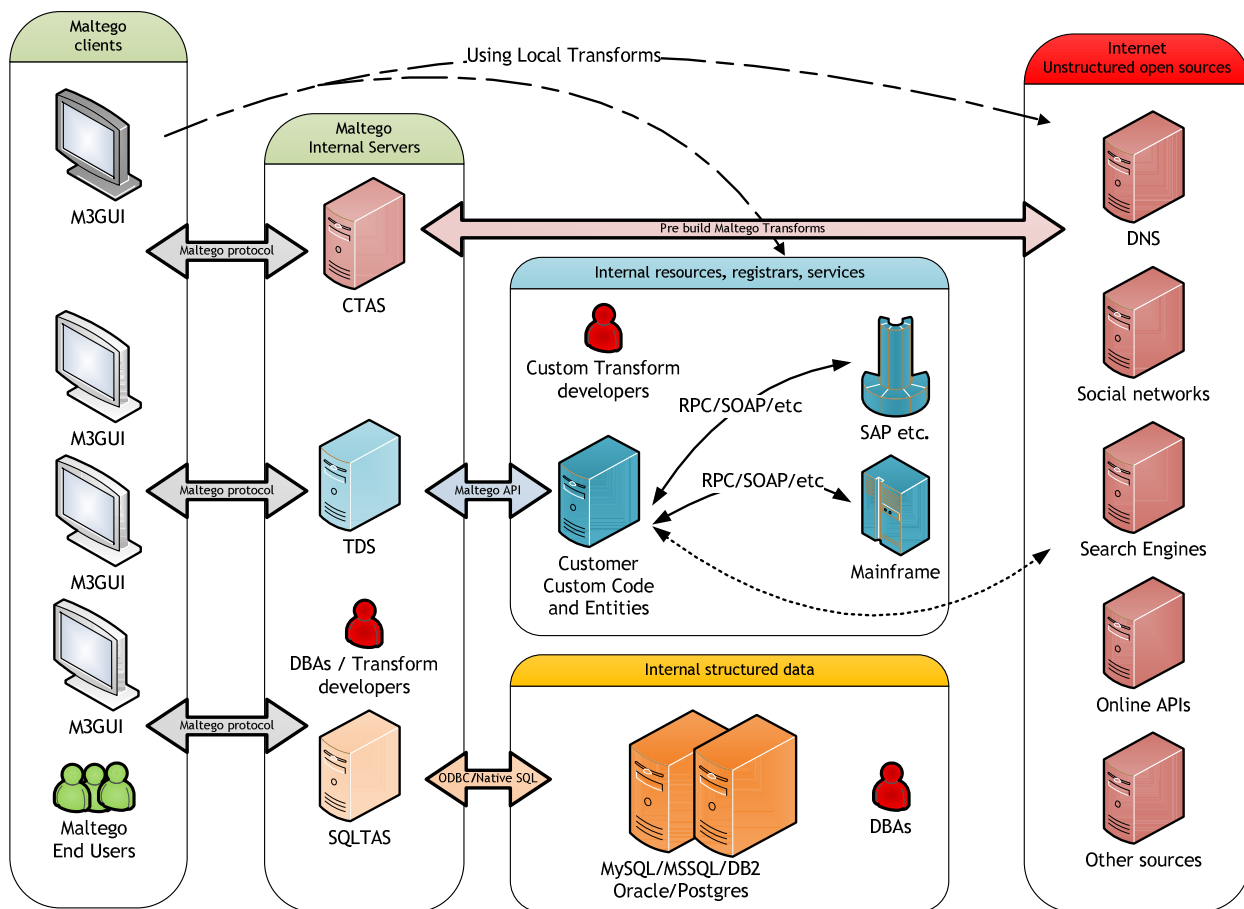
Introduction

The past the Maltego could only be extended with what is called 'Local Transforms'. These were pieces of code that were registered in the Maltego GUI and used the existing Maltego entities. The problem with these local transforms was that every user that wanted to utilize the transform had to have the code and environment on his or her physical computer. Furthermore it offered almost no code protection for the transform writer (as most of the transforms were written in scripting languages). As such it did not appeal to everyone and its community penetration was fairly low.

This situation has changed dramatically with the introduction of new technological advancements.

- **Custom entities:** Maltego version 3 gives the user the ability to create fully custom entities.
- **SQLTAS:** Server component that allows the user to write transforms using standard SQL queries and mapping language, and share amongst other Maltego users.
- **TDS:** Transform Deployment Server allows Maltego users to manage, share and control custom written transforms.

Combined this provides the user unheard of flexibility and power. The diagram below serves as an overview of the different technology components:



A typical integrator would want to deploy technology in the following areas:

Integration options with Maltego v3

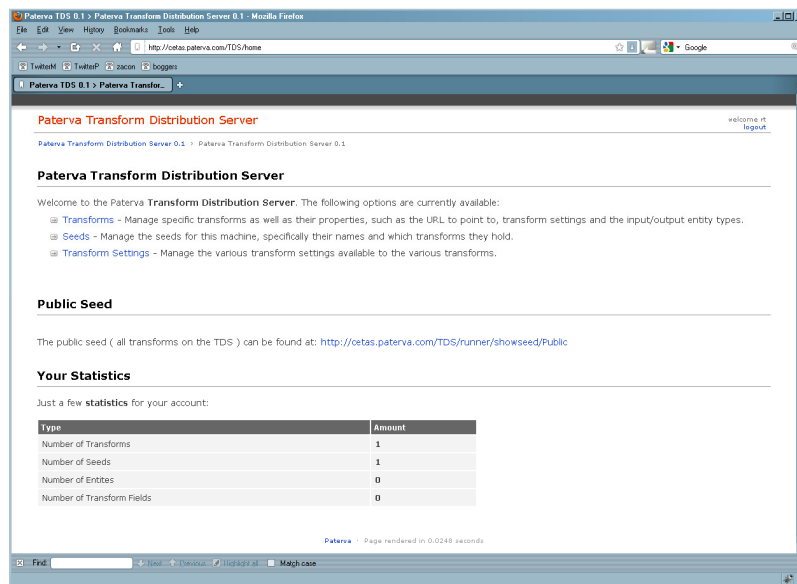
1. Local structured data – e.g. databases
2. Local services – e.g. SAP, services available on mainframe etc.
3. Unstructured data available on the Internet – e.g. search engines, social networks

TDS server

The Transform Deployment Server (TDS) is used to share custom written transform with other users. There are two versions of the TDS available:

- **cTDS:** Located on the Internet and open to all users. Transforms in the public seed is moderated.
- **iTDS:** Located on a customer's internal network. A single user administrates the server and there is no moderation on transforms created.

Once registered on the system (if using the cTDS, or logged in using the iTDS) the following screen is presented:



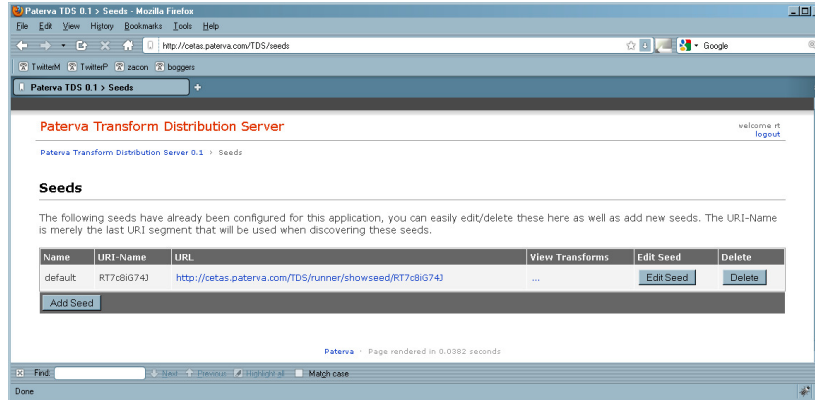
There are three main sections:

- **Transforms** – This is where the transforms are defined.
- **Seeds** – Used to cluster transforms together. The seed is all that need to be shared with external users.
- **Transform settings** – Transforms can use transform settings which is configured here.

Seed management

Seeds contain transforms and is used in the Maltego GUI to discover each transform with their settings, metadata etc. The first step when building transforms with the TDS is to create a seed. When logged in the first time, a default seed is available, but this can be edited:

Integration options with Maltego v3



Do not share this URL with anyone unless you want them to discovery your transforms. You could have as many seeds as you like. When editing a seed you can select which transforms should be contained in the seed. Do not worry about this if you haven't created transforms yet – you can add them later.

Transform settings

The next step is to create transform settings. This is optional. If your transform needs to get information from the GUI (from the user, not the selected entity) you would want to create a transform settings. In version 3 of Maltego these settings can also be marked as 'pop-up' – this means that the user will be prompted to enter a value in a pop-up before the actual transform runs. There are a couple of fields that you need to fill out. Give the setting a proper name and something useful in the Display and Default Values:

Add Transform Setting

Fill in the following form to edit this transform setting.

| | |
|---------------|---|
| Name | <input type="text" value="Age"/> |
| Display | <input type="text" value="Age"/> |
| Default Value | <input type="text" value="25"/> |
| Optional? | <input type="checkbox"/> False |
| Popup? | <input checked="" type="checkbox"/> Yes |

When done click 'Add Transform setting' – this will save the setting.

Transform Settings

The following transform settings have already been configured for this application, you can easily edit/delete these here as well as add new transform settings.

| Name | Type | Display | Default Value | Optional | Popup? | Edit Transform Settings | Delete |
|------|--------|---------|---------------|----------|--------|---|---------------------------------------|
| Age | string | Age | 25 | False | Yes | <input type="button" value="Edit Transform Setting"/> | <input type="button" value="Delete"/> |

This setting can now be used in any of the transforms.

Integration options with Maltego v3

Transforms

This is the meat of the application. In this section the transforms are configured. When adding a new transform you need to complete some forms.

The first three forms specify the transform name (this is how it will show up in the Maltego GUI's context menu), the Transform URL (more on that later) and the Input entity type. The entity type can either be selected from the drop down (standard Maltego entities) or manually entered. This is useful when you want your transforms to run on custom entities.

| | |
|----------------|--|
| Transform Name | <input type="text" value="MyTestTransform"/> |
| Transform URL | <input type="text" value="http://alpine.paterva.com/cgi-bin/testPy.py"/> |
| Input Entity | <input type="text" value="maltego.Phrase"/> |

Next you may fill in metadata. On the cTDS some of this information is pre-populated and cannot be changed:

| | |
|------------------------|--|
| Owner | RT |
| Disclaimer | <p>Please note the disclaimer will automatically include the following text:</p> <div style="border: 1px solid gray; padding: 5px;"><p>Please note this transform is being run on the Paterva Transform Distribution Server and has been written by the user 'rt'. This transform will be run on YOUR_URL_HERE and Paterva cannot be held responsible for any damage caused by this transform, you run this AT YOUR OWN RISK.</p></div> <p>For more information on this transform feel free to contact roelof.temmingh@gmail.com</p> |
| Description (optional) | <input type="text"/> |
| Version (optional) | <input type="text" value="1"/> |
| Author | roelof.temmingh@gmail.com |

Note that for security reasons a standard disclaimer is added to every transform served from the TDS. Lastly the Debug, Seed and Transform settings needs to be filled out:

| | |
|--------------------|--------------------------------------|
| Debug | <input checked="" type="checkbox"/> |
| Public | Not in public seed |
| Transform Settings | <input type="text" value="Age"/> |
| Seeds | <input type="text" value="default"/> |

The 'AskAge' transform setting that was defined in the previous section is available and selected. The selected seed is 'default' – the default seed. You may choose any number of seeds or settings using control left click.

Integration options with Maltego v3

The TDS will do some cursory syntax and make sure that the URL returns a HTTP 200 code before accepting the transform. If everything checks out OK the transform will be added to the seed:

Successfully Inserted 'MyTestTransform'

(cTDS only) If you want to share this transforms with all Maltego users can you choose to submit it for moderation:

Public Seed

Please select the transform(s) you would like to be available in the public seed. This is the seed that will be available to all commercial and community Maltego clients.

All transforms specified to be in the public seed will need to be moderated and as such please make sure of the following:

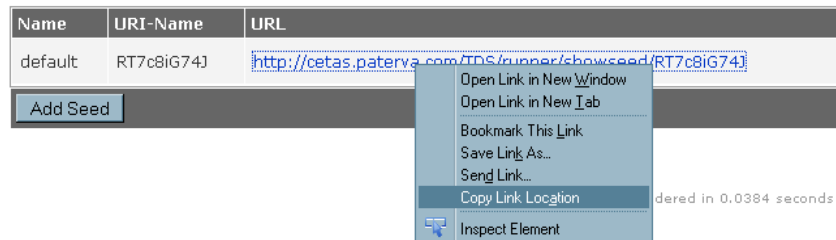
- You have completed all the meta-data (description, disclaimer, version, etc)
- Your transform **works** (test them via your private seed)
- If you use any custom entities you have included a sample graph with the entities or the exported entities somewhere

| Name | URL | Moderation Status | Add for Moderation | Remove from Moderation |
|-----------------|---|-------------------|--------------------|------------------------------------|
| MyTestTransform | http://alpine.paterva.com/cgi-bin/testPy.py | Private Transform | Add to Public Seed | Remove from Public Seed/Moderation |

If you are sure your transform is working and it should be shared you can click on 'Add to Public Seed' and it will be added to the public seed if moderated.

Discovering from the Maltego GUI

Start the Maltego GUI. Go to the Manage tool bar. Click on Discover Transforms. On the TDS navigate to the Seed section. Right click on the Seed URL and copy the link location to the clipboard:



Enter the seed into the URL text box in 'Discover Transforms' wizard, give it a useful name and click on Add.

Integration options with Maltego v3

Maltego GUI

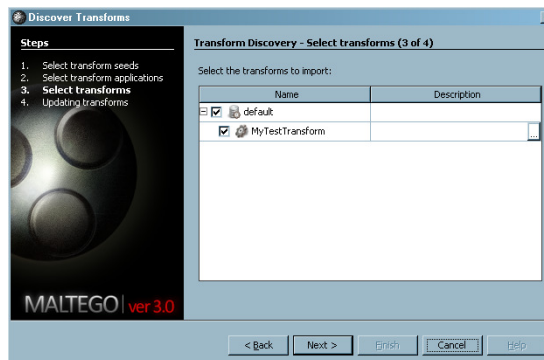
| Name | URL | Description |
|--|---|-------------|
| <input checked="" type="checkbox"/> CTAS 3 | https://alpine.paterva.com/CTAS3.xml | ... |
| <input checked="" type="checkbox"/> TDS | http://cetas.paterva.com/TDS/runner/showseed/RT7c8IG74J | ... |

TDS web interface

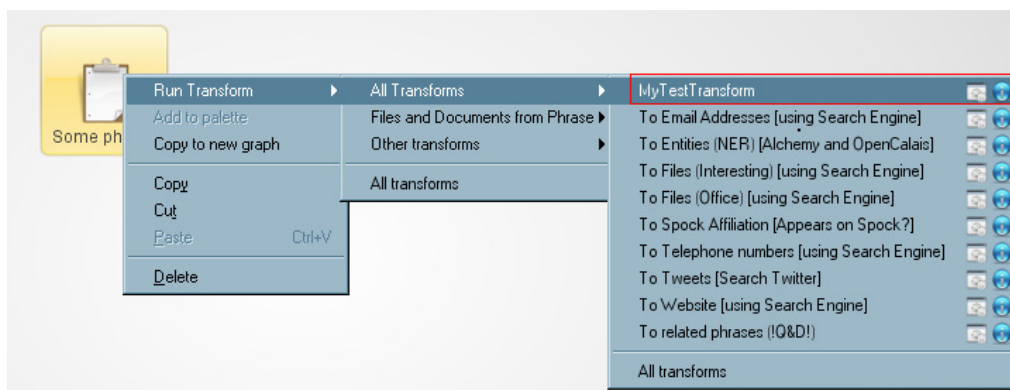
| Name | URI-Name | URL |
|---------|------------|---|
| default | RT7c8IG74J | http://cetas.paterva.com/TDS/runner/showseed/RT7c8IG74J |

For the diagram above it should be clear that the seed used in the GUI is the same as in the TDS web interface.

Follow the wizard and you'll see that the 'MyTestTransform' has been discovered:

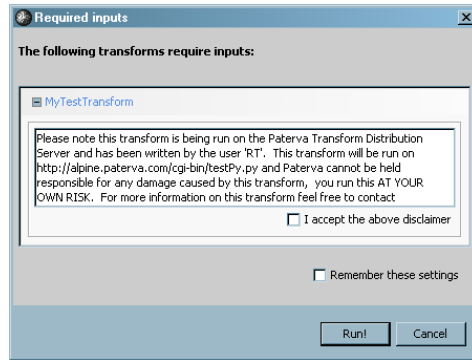


As the transform is configured to work on a Phrase entity we can now drag a Phrase entity from the palette and the right click context menu should display our transform (marked in red below):

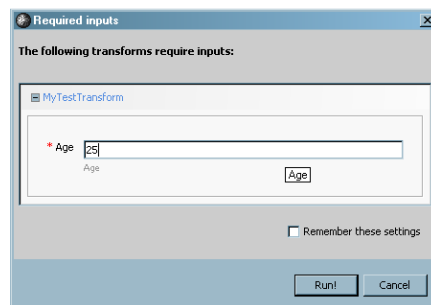


Running the transform will reveal the disclaimer:

Integration options with Maltego v3



If the transform setting is marked as 'popup' (this still has to be done manually at this stage) the popup will now appear:



Note that the popup corresponds with the transform setting created within the TDS.

The transform in code

This is probably the most important part of the process – the transform code itself. The URL that was specified when the transform is created points to a CGI where the real work is done. This CGI can live on any system, internal or on the Internet and can be coded in any language, as long as it returns output in a standard way.

If the URL is defined as **http://server/script** it is called as follows:

```
POST /script?value=XXXX&fields=YYYY
<XML>
</XML>
```

Where XXX and YYY is the base64 encoded parameters as follows:

- XXX:value of entity
- YYY: fieldname1=fieldvalue1#fieldname2=fieldvalue2

The CGI parameters are passed to make it easy to extract the values quickly without having to parse the XML. Because HTTP parameters are limited to 255 characters you might not get all information here. The better option is to parse the XML.

Integration options with Maltego v3

The XML is the full Maltego Specification 2 of the message and means that the developer can obtain things like the slider value, the weight of the entity, transform settings as well as all additional fields. In order to make writing transforms easier Python and PHP libraries are available. Other development platforms might follow.

The Maltego library is used to parse the incoming XML and generate the outgoing XML. There are 3 classes in the Maltego module:

MaltegoEntity class:

Used to construct a Maltego entity to be returned.

Methods:

- `__init__(self,eT=None,v=None):`
- `setType(self,eT=None):`
- `setValue(self,eV=None):`
- `setWeight(self,w=None):`
- `setDisplayInformation(self,di=None):`
- `addAdditionalFields(self,fieldName=None,displayName=None,matchingRule=False,value=None):`
- `setIconURL(self,iU=None):`
- `returnEntity(self):`

MaltegoTransform class:

Used to send constructed Maltego entities back to the client.

Methods:

- `addEntity(self,enType,enValue):`
- `addEntityToMessage(self,maltegoEntity):`
- `addUIMessage(self,message,messageType="Inform"):`
- `addException(self,exceptionString):`
- `throwExceptions(self):`
- `returnOutput(self):`
- `debug(self,msg):`

MaltegoMsg class:

Used to read incoming Maltego POST XML and break up into usable variables

Methods:

- `__init__(self,MaltegoXML=""):`

Typically these classes would be used in a Python script as follows:

Integration options with Maltego v3

testPy.py (make sure it's set to be +x and in a directory where it executes and which is part of your webroot – e.g. a CGI-BIN directory)

---cut here--->

```
#!/usr/bin/python
#           ^ Or where ever your Python lives (type 'which python')
import sys
import os

# Importing from the Maltego.py module
from Maltego import *

# Put this here so that we can catch errors
sys.stderr = sys.stdout

def main():
    # Got to set content type as we are returning XML to the POST
    print "Content-type: xml\n\n"

    # Read raw POST from STDIN -
    # note that you dont need to use the Python CGI Libs
    MaltegoXML_in = sys.stdin.read()

    # Create a object called 'm' that will contain all the parsed XML info
    m = MaltegoMsg(MaltegoXML_in)

    ##> STARTCOMMENT
    # If transform is marked as debug - this will show in client
    # To disable, comment up to ENDCOMMENT

    # m.Value = Value of the selected entity
    # m.Type = Type of the selected entity
    # m.Weight = Weight of selected entity
    # m.Slider = The position of the slider in the GUI (12,55,255,10 000)

    print 'Type='+m.Type+' Value='+m.Value+' Weight='+m.Weight+' Limit='+ m.Slider

    # m.AdditionalFields directory - e.g. m.AdditionalFields["MyField"]
    print '\nAdditional fields:'
    for item in m.AdditionalFields.keys():
        print 'N:'+item+' V:'+m.AdditionalFields[item]

    # m.TransformSettings directory - e.g. m.AdditionalFields["MyField"]
    print "\nTransform settings:"
    for item in m.TransformSettings.keys():
        print "N:"+item+" V:"+m.TransformSettings[item]

    ##< ENDCOMMENT

    # Now we have all the data - lets write a transform!
    Age="0"
    if m.TransformSettings["Age"] is not None:
        Age = m.TransformSettings["Age"]

    # Create MaltegoTransform object
    TRX = MaltegoTransform()

    # Add entity with Type and Value
    Ent=TRX.addEntity("maltego.Person","doesnotmatter_its_computed")

    # Set the weight
    Ent.setWeight(99)

    # Add additional fields
    # In this case we are swapping the first name and the lastname...
    Ent.addAdditionalFields("firstname","First Names","strict",m.AdditionalFields["lastname"])
    Ent.addAdditionalFields("lastname","Surname","strict",m.AdditionalFields["firstname"])
    # ...and adding an Age field that comes from transform settings
    Ent.addAdditionalFields("Age","Age of Person","strict",Age)

    # Set IconURL from transform setting
    if m.TransformSettings["ImageURL"] is not None:
        Ent.setIconURL(m.TransformSettings["ImageURL"])
```

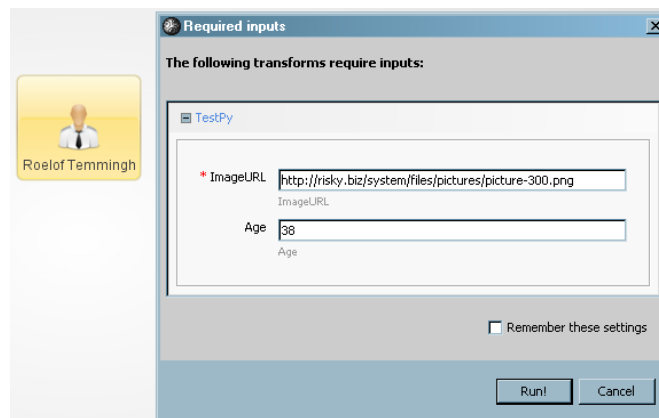
Integration options with Maltego v3

```
# Return the output XML
TRX.returnOutput()
```

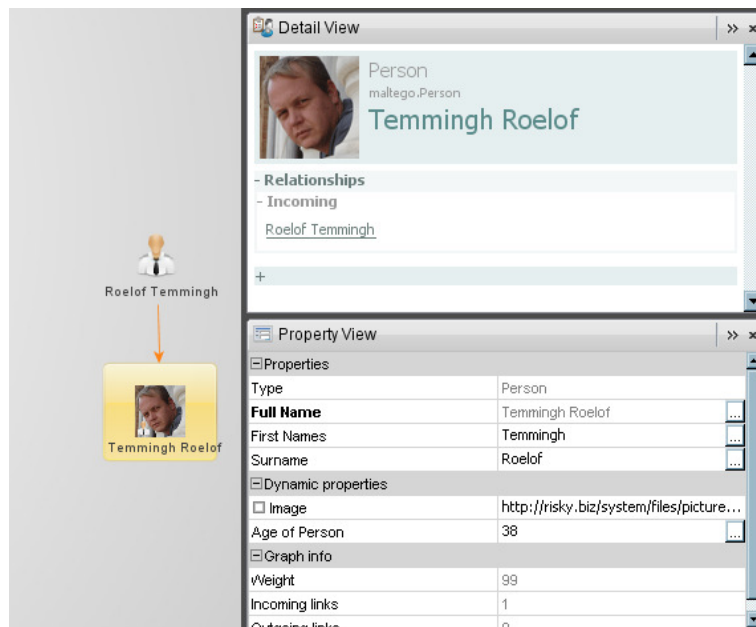
```
##
main()
```

This particular transform will take a Person Entity as input, reverse the first name and last name and add an additional field called 'Age' to the entity (which is matched strictly). This value is obtained as a popup from the GUI. It also looks for another transform setting called 'ImageURL' and adds that to the output entity so that the user can set the icon URL of the entity (note that in the above example this additional field was not created).

Let's see how this works in the Maltego GUI:



This results in the following graph:



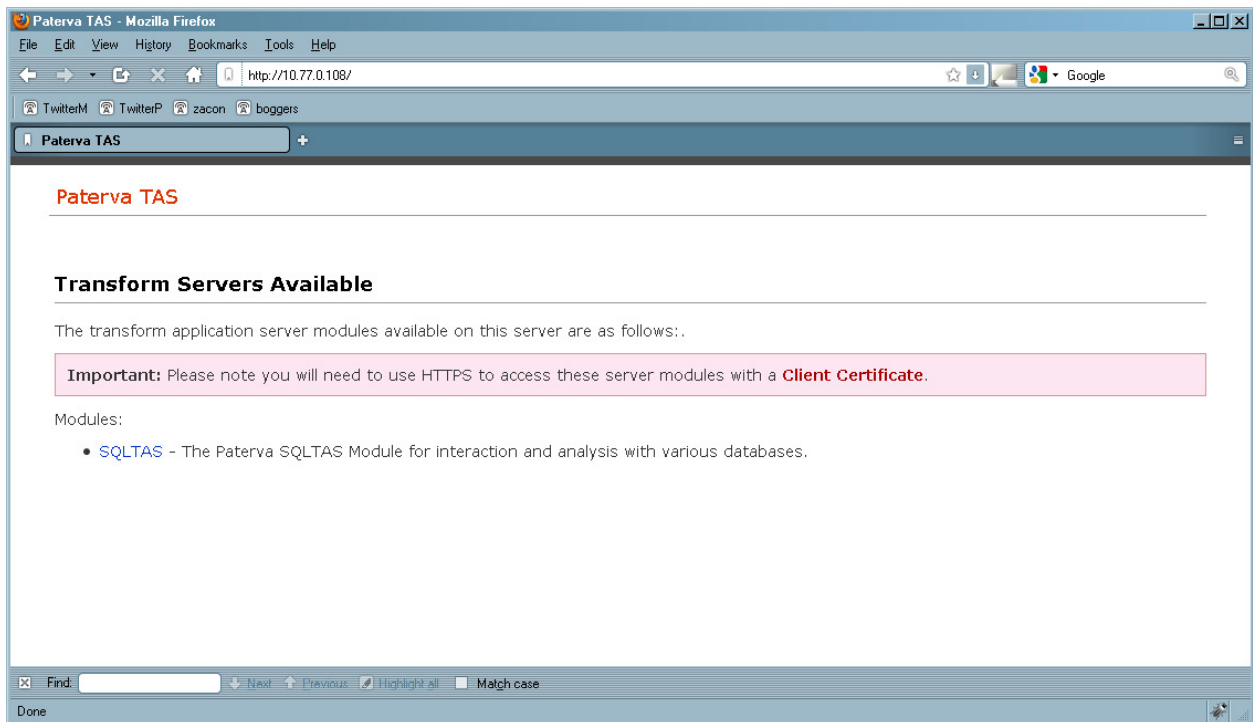
Integration options with Maltego v3

As can be seen in the diagram above – the new entity has its First name and Surname reversed, has an additional field called ‘Age’ that’s displayed as ‘Age of Person’. Also - the IconURL has been set and the weight is set to 99.

SQLTAS

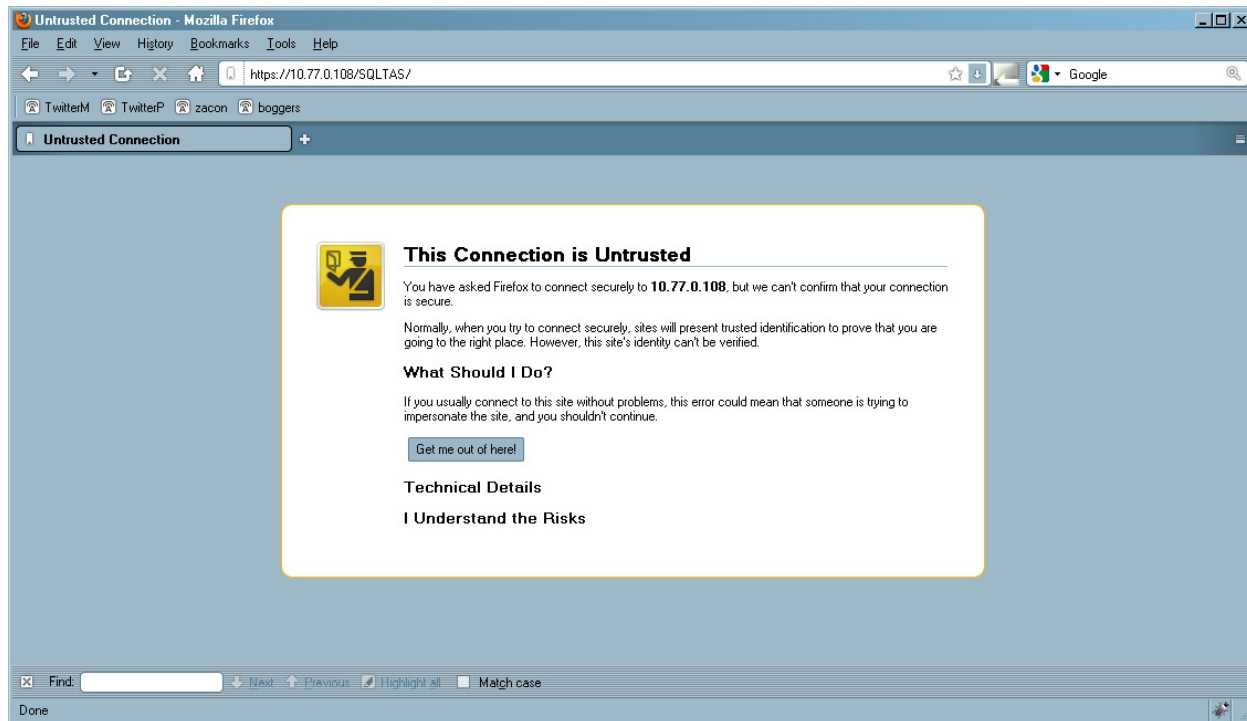
Connecting to the server with a browser:

Once you have configured your SQLTAS to have an IP address and installed the client certificate (provided to you by Paterva) you can connect to it with a browser. Open the browser and surf to <http://<<your server's IP address here>>>. You should see the following:



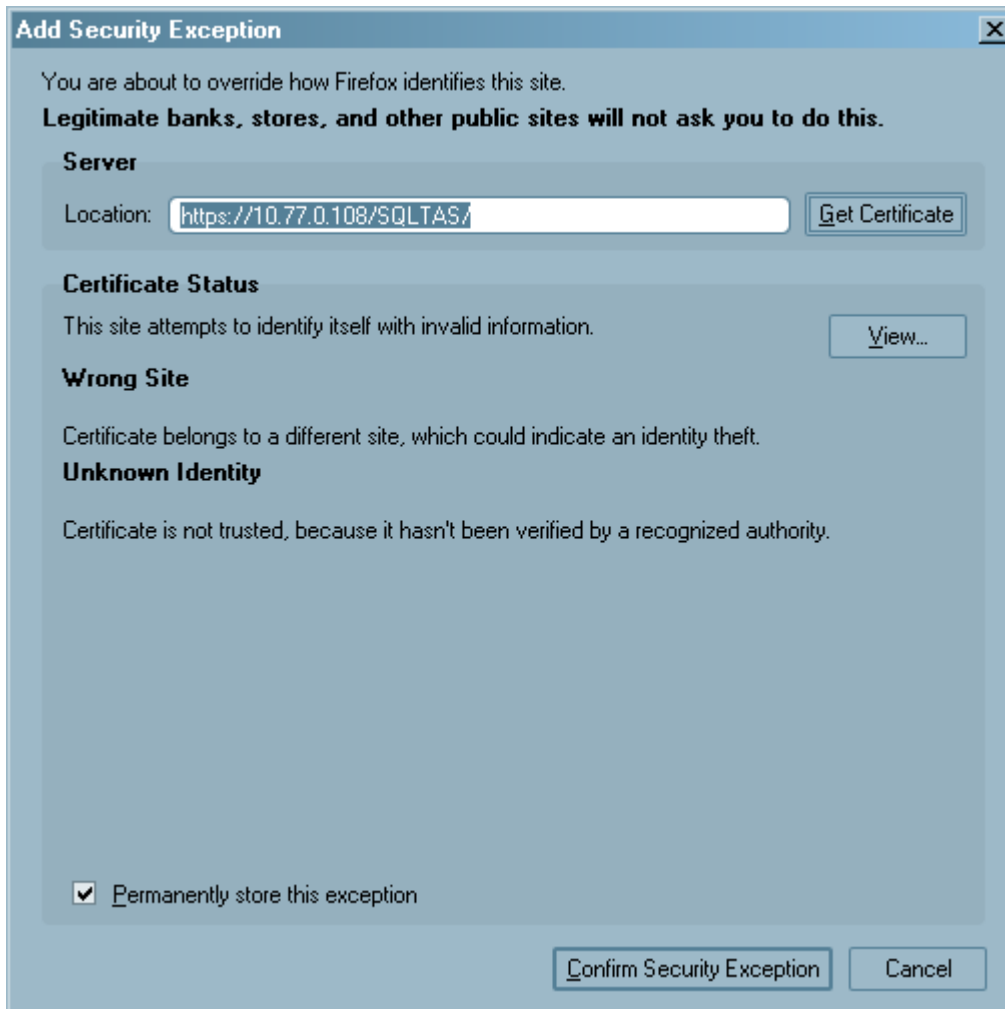
This is the only screen that is visible from the open HTTP server. The SQLTAS configuration is all done over encrypted HTTPS. Click on the link that read ‘SQLTAS’. Because the server’s certificate is signed by Paterva’s CA (and your browser does not know it) you’ll get a certificate warning:

Integration options with Maltego v3



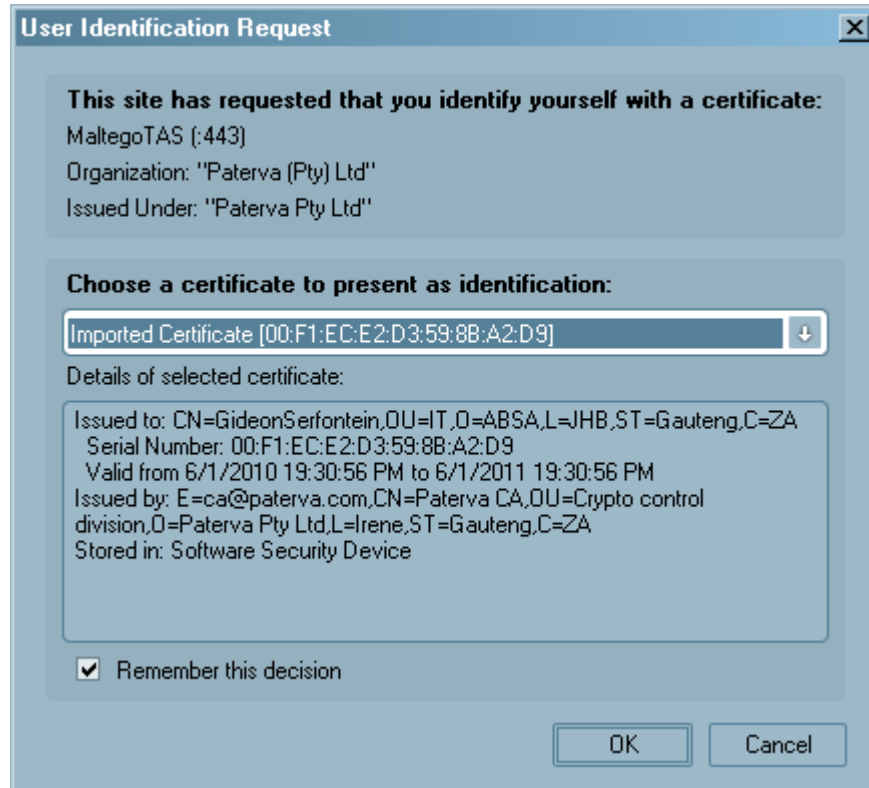
You can safely ignore this warning (or install the Paterva CA certificate in your browser if you feel like it) by clicking on 'I understand the Risks', 'Add exception' and finally 'Confirm exception'. In IE this messages looks different (but you know the drill by now):

Integration options with Maltego v3

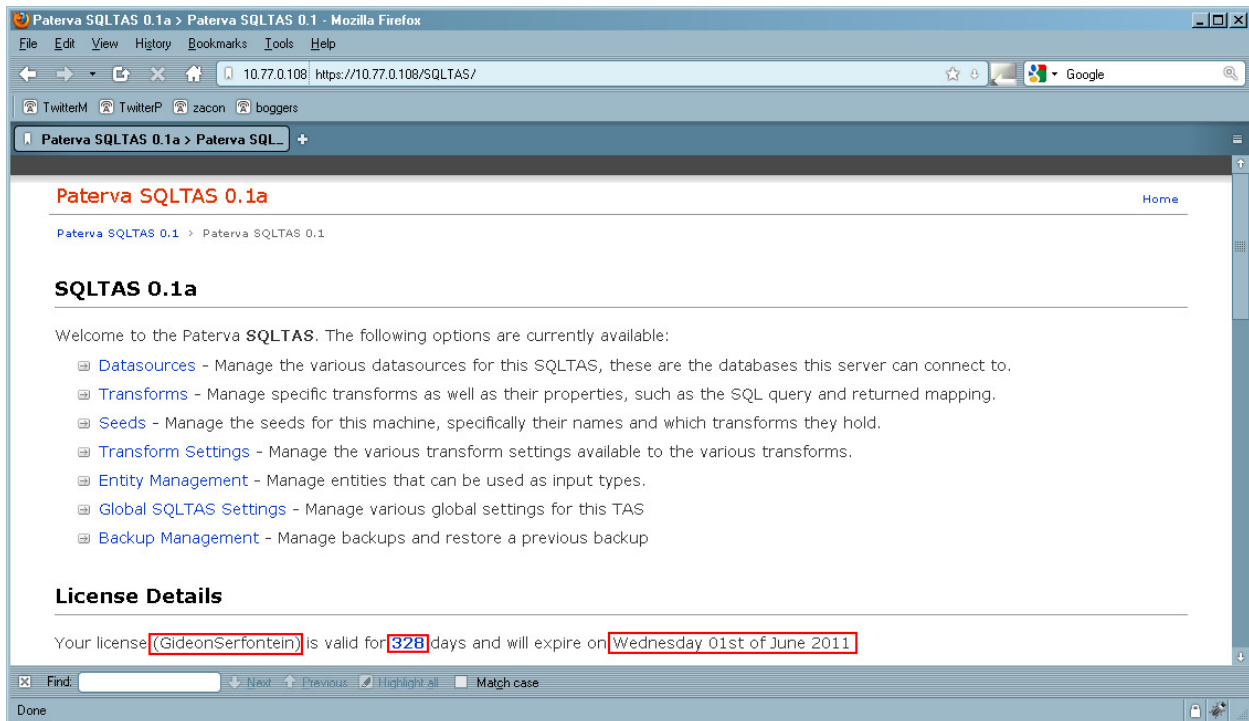


The next screen will ask you which client certificate you'd like to present:

Integration options with Maltego v3



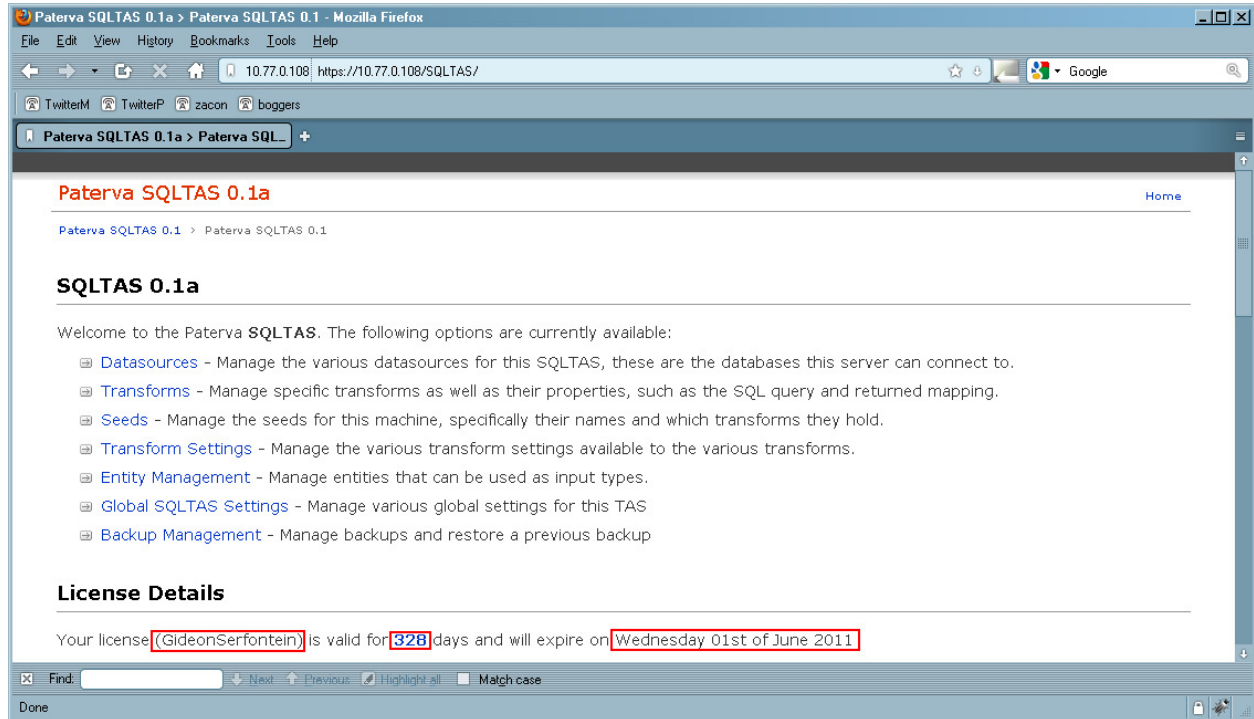
If you have multiple client certificates installed in your browser, you should choose the one that was issued to you by Paterva. Click on OK and you should get to the SQLTAS interface:



Integration options with Maltego v3

The first time you connect to the SQLTAS interface your client certificate ‘bonds’ with the server – the server will now identify your client certificate every time and keep track of the license. This procedure works similarly within Internet Explorer.

Once logged into the SQLTAS the following menu items are available:



Procedure

The basic procedure of setting up the SQLTAS is as follows:

1. **Define one or more data sources.** Here you can select the database type, host, credentials, and in some cases the database name. The data source is given a name that is used in the next sections.
2. **Define seeds.** This is a URL that will be used in the Maltego client for discovering your transforms. Transforms are ‘packed’ into a seed. A single seed can contain more than one transform and two unique seeds can both contain the same transforms.
3. **Define transform settings (if needed).** In some cases you’d want to get input from the user before the transform runs. This is done either by pop-ups or within the transform manager (on the GUI client). These values can be used within SQL statements.
4. **Define transforms.** Finally you can set up transforms – for this you will use the Data Sources as defined in step 1, packaging them into seeds defined in step 2 and, if you like, using transform settings as defined in step 3. The syntax for reading and writing entity values, types, display values, transform settings and so on is explained in the next section.

Integration options with Maltego v3

The order of these steps are no important – that is – you can define more seeds later or add transforms settings as you go along, but for first time use it makes sense to follow them as they provide a logical order.

Mapping language

The following is a concise guide to reading and writing entities in the SQLTAS – it can also be found when clicking on the '?' in the interface itself.

Reading Entities from the Maltego client (SQL query)

The following table shows the values that can be read from the incoming entity and how it's used within a SQL query:

| | |
|---------------------------------|--|
| E.value | The value of the entity. |
| E.field('fieldname') | The value of the entity field called 'fieldname'. |
| E.weight | The value of the weight of the entity. |
| T.setting('settingname') | The value of a transform setting. These are defined in the transform settings section and are per transform, not per entity. |

Typical use within a SQL query:

```
SELECT username,address,phone FROM table
WHERE username = '$$E.value$$' AND phone LIKE '%$T.setting('phone')$$%'
```

Note that **everything** between the tokens '\$\$' are seen as Maltego inputs.

Writing entities to the Maltego client

Mapping the SQL output to nodes in Maltego is done as follows:

```
{node1}{node2}..{nodeN}
```

Where a node is defined as follows:

```
{NodeParam1 = 'Value1'; NodeParam2 = 'Value2'; ...NodeParamN = 'ValueN';}
```

The following parameters can be set:

| | |
|-----------------------------|---|
| E.type | The entity Type. This have to correspond with the Maltego type defined in the client application (GUI). |
| E.value | The value of the node. |
| E.field('FieldName') | Setting a field within the entity. |

Integration options with Maltego v3

| | |
|-----------------------------------|---|
| E.weight | The weight value of the entity. |
| E.displayInfo('LabelName') | The display value for label with name 'LabelName' |

There are two ways of setting values for parameters:

\$0...\$N

Corresponds to the column number of resultant SQL query. **Numbering starts at zero (0).**

\$columnName

Where **columnName** is the name of the column corresponding to the SQL query.

Thus, typical use – assume a query as follows:

SQL query:

```
SELECT registration,vin,owner,SSN FROM cars WHERE owner = '$E.value$'
```

Mapping:

```
{E.type = CarEntity; E.value = '$registration'; E.field('vin') = '$vin';}
```

```
{E.type = Person; E.value = '$owner'; E.field('SSN') = '$SSN';}
```

This would result in 2 types of entities – a CarEntity (if defined within the client), and a Person entity. The same mapping could also be written with column numbers as follows (note that numbering starts at zero):

```
{E.type = CarEntity; E.value = '$0'; E.field('vin') = '$1';}
```

```
{E.type = Person; E.value = '$2'; E.field('SSN') = '$3';}
```

The minimum requirement for an entity is a type and a value. As an example:

SQL query:

```
SELECT phone FROM mytable WHERE name = '$E.value$'
```

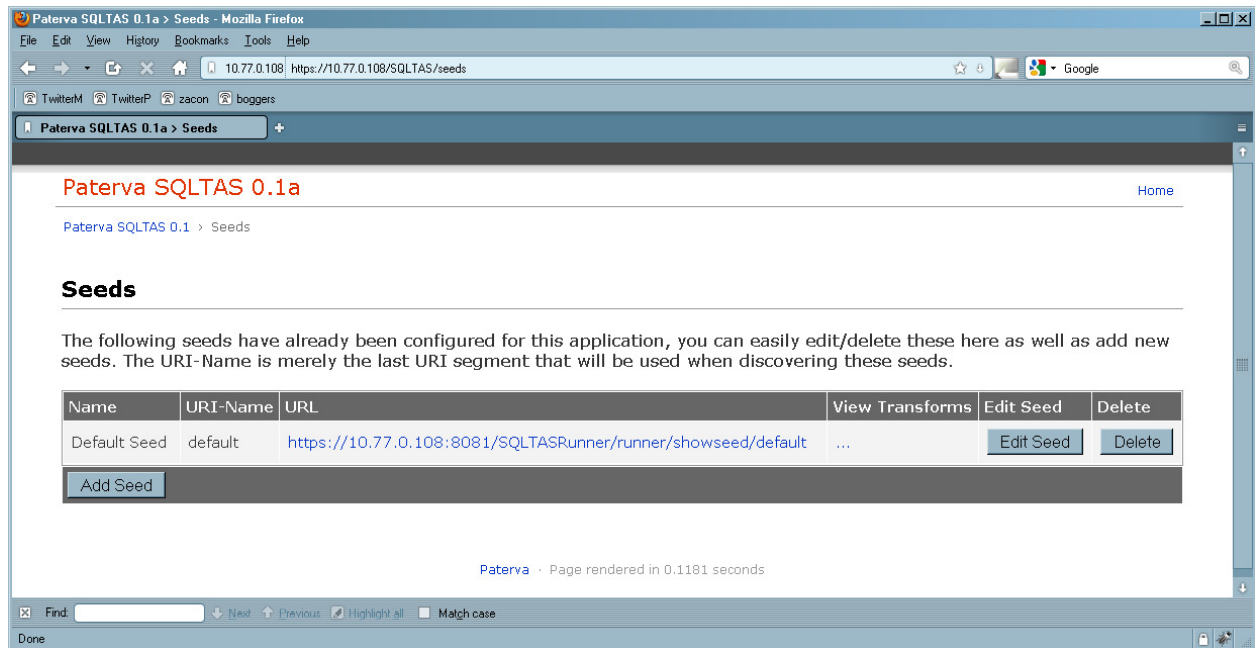
Mapping:

```
{E.type = PhoneNumber; E.value = '$0';}
```

Running a transform via the SQLTAS on the client

The first step here is to discover the seed as defined in the web interface. Let's assume the seed is as follows:

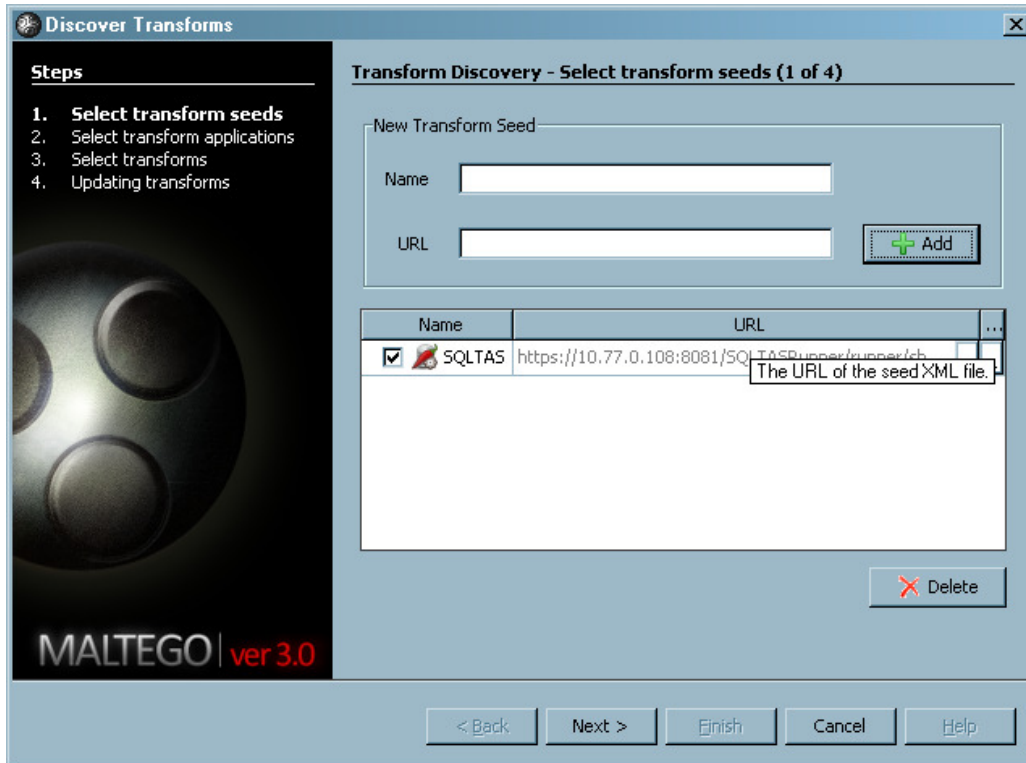
Integration options with Maltego v3



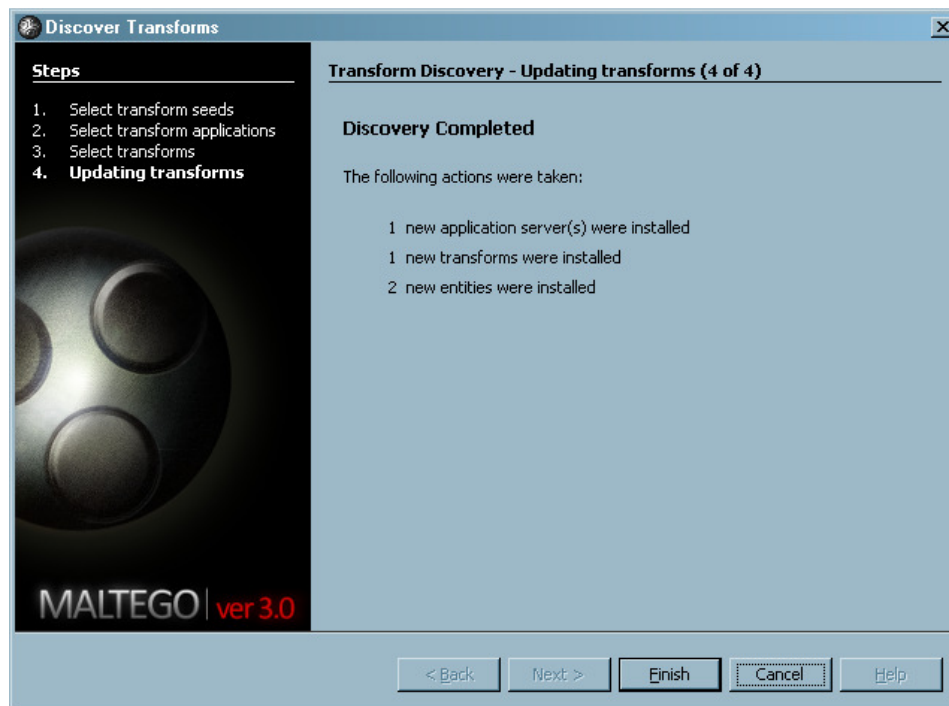
Using the Maltego the client we'll do the following:

- Open GUI
- Click on **Manage** in the Ribbon
- Click on **Discover** Transforms
- In the **Name** text field add 'Local SQLTAS' (or anything that will describe it to you)
- In the **URL** text field enter the URL from the web interface
- Click on the [+] button

Integration options with Maltego v3



- Click on **Next** and follow the wizard up to the last screen:



The Maltego client will now tell you how many transforms were discovered (in this case just the one default transforms)

Integration options with Maltego v3

At this stage the transforms discovered from the SQLTAS is loaded in the client. Clients should rediscovery from this seed under these conditions:

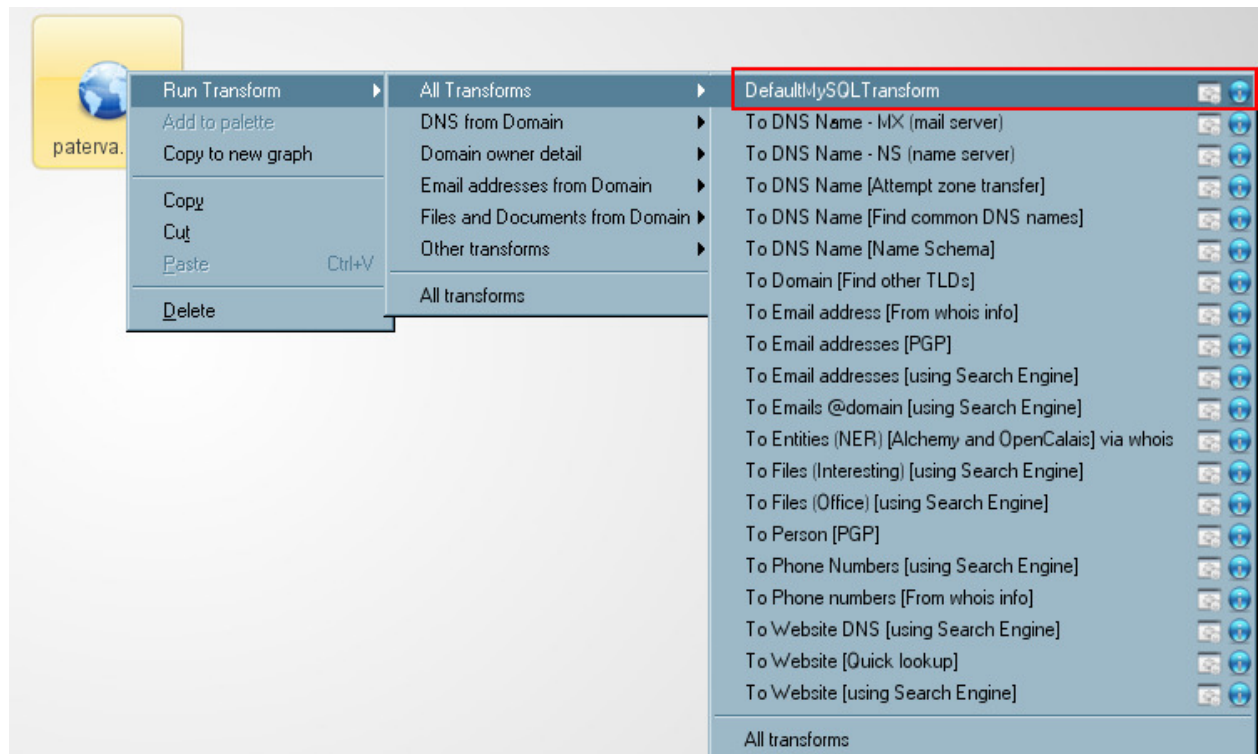
- New transforms added
- Transforms contained in seed changed
- Transform settings changed, or settings per transform
- Entity INPUT type changed

Rediscovery is **NOT** needed in the following situations:

- SQL statements changes
- Data source changes
- Entity Mapping changed
- Entity OUTPUT mapping changed

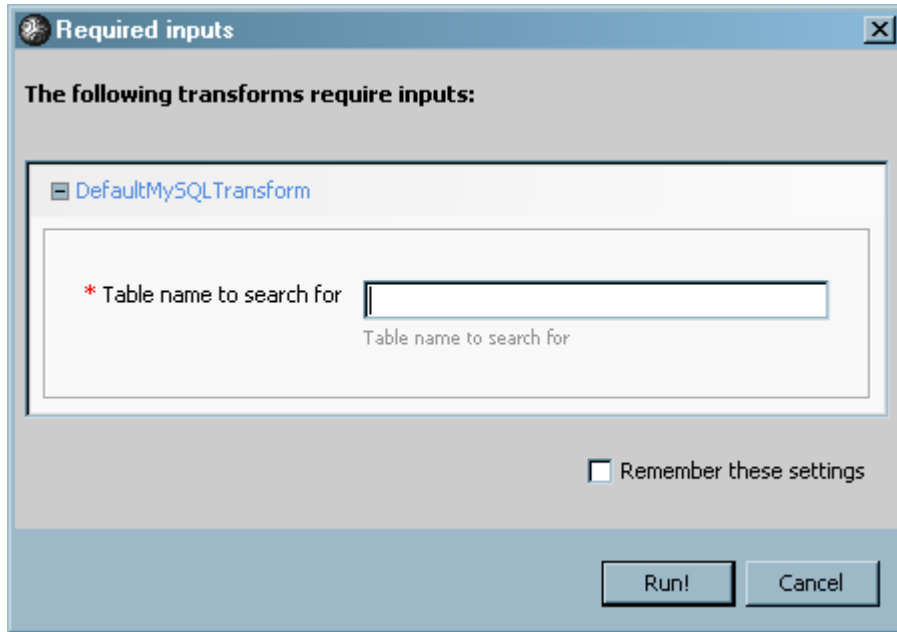
Running the discovered transform

If we now add a domain to the graph (since the example transform uses domains as input) and right click we can see the 'DefaultMySQLTransform' in the context menu item:



This transform uses transform setting – which pops up right after we click on the transform:

Integration options with Maltego v3



The SQL statement runs on the *information_schema* table (default MySQL table) and reads:

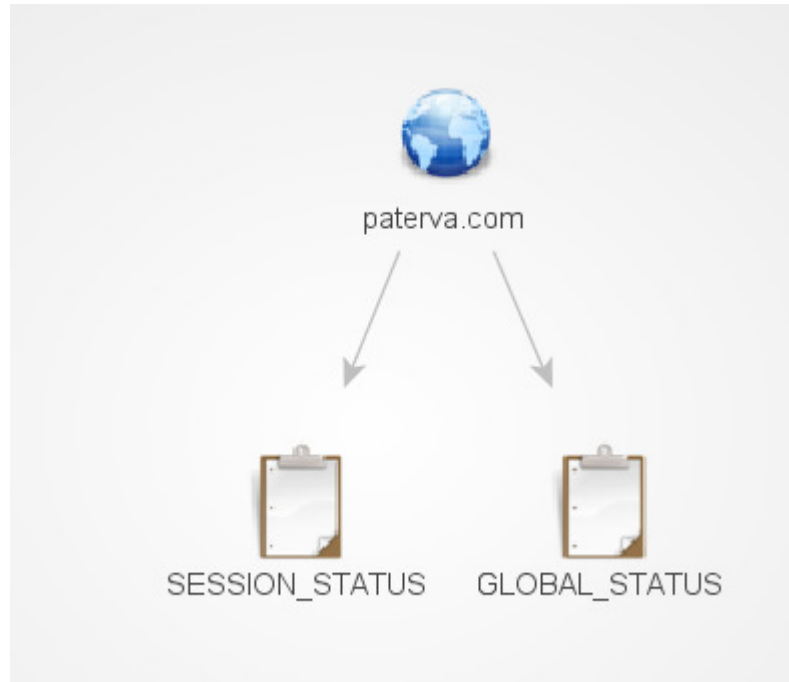
```
SELECT TABLE_NAME FROM TABLES where TABLE_NAME Like  
'%$$T.setting('Table')$$%'
```

The mapping read:

```
{E.type=Phrase;E.value='$0';}
```

Thus - if we enter the word 'STATUS' in the pop-up we get the following graph:

Integration options with Maltego v3



Note that entities are of type Phrase and their value corresponds to the first column in the returned table.