API Übersicht

Was ist die FLOWFACT API?

Die FLOWFACT API ist ein RESTful Web Services der den Zugriff auf die Datenbank der FLOWFACT Produkte bietet.

Diese Schnittstelle ermöglicht es Kunden und Partnern individuelle Lösungen gegen FLOWFACT Produkte zu entwickeln. Ob Schnittstellen zur Warenwirtschaft, Darstellung von Daten auf der eigenen Homepage oder komplett eigene APPs. Über die FLOWFACT API ist es möglich Daten direkt aus dem FLOWFACT System auszulesen und in diesen Bereichen darzustellen.

Was liefert diese Übersicht?

Diese Dokumentation liefert eine Anleitung zur Installation und Nutzung der FLOWFACT API. Wir möchten gerade externen Entwicklern hier eine Möglichkeit bieten, sich in die Funktionsweise der FLOWFACT API einzulesen und die notwendige Übersicht über Ressourcen und Funktionsweisen zu bekommen.

Was diese Übersicht nicht liefert

Zur Nutzung der FLOWFACT API benötigen Sie Kenntnisse bei der Nutzung von RESTful Web Services. Je nachdem aus welchem Bereich dieser Aufruf erfolgt benötigen Sie Wissen um die entsprechende Programmiersprache wie C#, PHP oder Java. Auch der Umgang mit XML Dateien sollte Ihnen vertraut sein und Kommunikation über das Internet aus Ihrem Programm stellt für Sie kein Problem da.

Diese Dokumentation richtet sich nicht an Anfänger und bietet keine Anleitung zum Programmieren!

Support wird nicht bereitgestellt

Sprache der Dokumentation
Bitte beachten Sie, dass die FLOWFACT API von einem internationalen Team entwickelt wird. Auch sind die Nutzer der FLOWFACT API nicht immer deutschsprachig. Aus diesem Grund sind einige Artikel dieser Seite, gerade im Bereich der Technik und des Programmierens, auf englisch geschrieben.

Häufig gestellte Fragen

- Ist die Nutzung der API kostenpflichtig?
- Wie installiere und richte ich die FLOWFACT API ein?
- Wie melde ich mich an einer installierten API an und führe erste Abfragen durch?
- Ich bin Entwickler und würde die API gerne testen, wie kann ich das tun?

Andere Ressourcen

- Ressourcenübersicht der aktuellen FLOWFACT API
- Lizenzierungsinformationen

Nach Thema durchsuchen

Kürzlich aktualisierte Artikel

Systemvoraussetzungen für Installation On-Premise Installationen (eigener, lokaler Server)
12.Feb.2019 • aktualisiert von verena baumhardt2 • Änderung anzeigen

API Übersicht
08.Feb.2019 • aktualisiert von verena baumhardt2 • Ände

A-H	I-Q	R-Z
api	installation	repair
asked	installieren	repairkit
characteristics	introduction	required
conventions	kit	servermanager
decisions	mappings	test
deployment	properties	tomcat
documentation	questions	update
frequently		

0-9
8

rung anzeigen

- Configuring FLOWFACT API and Mobile in a DMZ (Demilitarized Zone)
  14.Nov.2017 • erstellt von Andreas Runschke
- Absicherung des Systems mit einem gekauften (und vertrauten) SSL Zertifikat
  21.Aug.2017 • aktualisiert von Anonym • Änderung anzeigen
- Erstinstallation auf Linux Systemen
  02.Aug.2017 • aktualisiert von Anonym • Änderung anzeigen
- Erstinstallationen
  30.Mai.2017 • aktualisiert von Anonym • Änderung anzeigen
- XML-Schema (XSD)
  01.Feb.2017 • aktualisiert von Andreas Runschke • Änderung anzeigen
- API Sandbox for Testing
  22.Sep.2016 • aktualisiert von Andreas Runschke • Änderung anzeigen
- API general information
  16.Sep.2016 • aktualisiert von Andreas Runschke • Änderung anzeigen
- Tutorials & Best Practices
  16.Sep.2016 • aktualisiert von Andreas Runschke • Änderung anzeigen

API für Entwickler (API for developer)

Sie sind Entwickler und ein Kunde hat Sie beauftragt über die FLOWFACT API etwas für Ihn zu programmieren? Eventuell wollen Sie auch eine Lösung auf eigene Faust entwickeln und diese danach allen FLOWFACT Kunden anbieten?

Sehr schön! Dieser Bereich ist nur für Sie. Hier finden Sie eine ausführliche Dokumentation der API Ressourcen und Beispielprogrammierungen um mit der Entwicklung beginnen zu können.

You are Developer and a customer has assigned you to develop some tool against the FLOWFACT API? Possibly you want to develop some tool on your own and sell it afterwards to all FLOWFACT customer?

Very nice! This area is just for you. You can find a detailed documentation of the API ressources and also Best Practice code snippets to directly start with developing.

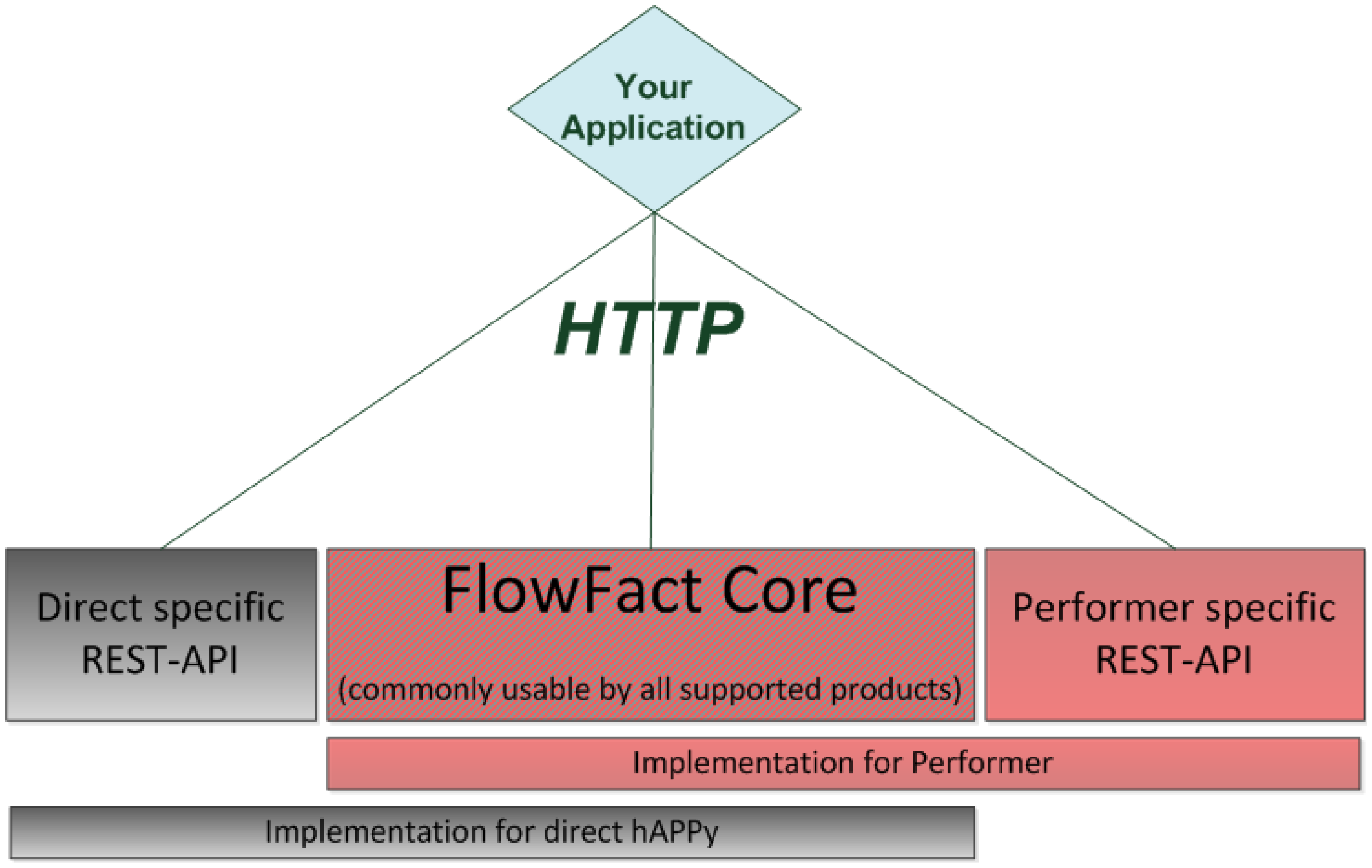
Die Themen (Topics)

API general information

The goal of the FlowFact REST-API is to provide an external access to internal data. These mechanisms can be used to Create, Read, Update or Delete data from / to the FlowFact system. One could say the REST-API implements a so called CRUD-Interface (**C**reate **R**ead **U**pdate **D**elete) while not every operation is available to every resource.

In order to enable the usage of this common interface to a wide variety of FlowFact products, a base RESTful API is going to be implemented which is supported by a variety of FlowFact products. In addition to this basic interface every single FlowFact productline may have its additional services that are not implmented on other products (because they do not make sense or the underlying functionality is not part of the product for example).

The following diagram shows the basic layout of the REST interface of the Product "Flowfact Performer" and "FlowFact direct hAPPy":


Your Application
HTTP
Direct specific REST-API
FlowFact Core
(commonly usable by all supported products)
Performer specific REST-API
Implementation for Performer
Implementation for direct hAPPy

As pointed out before in the middle of this schema one can see the basic interface ("Core") - it is implemented by both product backends and enables the user of the REST-API to perform CRUD operations on users, contacts, inquiries, activities, etc.. On the right side special functions for the FlowFact Performer are indicated by the overlapping red box (backed up by the corresponding implementation), on the left side direct hAPPy specific functionality is being shown. It might very well be possible that product specific interfaces are being merged to the core specification in the future and thus have to be implemented by every productline which provides the basic REST interface.
When using Version 1.0 of the REST-API there is no difference between the abilities of FlowFact direct hAPPy and Performer.

Basic layout - What are the available resources in every FlowFact System?

FlowFact itself is a highly customizable CRM system in which almost every customized workflow may be implemented. That means that, for example, on System A the contents of a specific Inquiry for "Appartment for rent" may differ from those of a System B, but they both have got the inquiry type "Appartment for rent". What would the message for "Create a new inquiry for an appartment for rent" look like on System A? And which information must be contained to be able to be created on System B? Well, for sure they differ because the available fields may differ. Now, keep in mind that one of the main goals of the REST API is to support each and every FlowFact that is currently deployed. This is why we are working on creating a so called "baseline" for the REST-API.

BASELINE - RESOURCES AND MESSAGING WITHING THE COMMON REST-API

Part of the baseline is, for example, that every FlowFact has users, inquiries, contacts, activities, etc.. Even some specific types are part of the baseline: "Appartment for rent" for example is one of the available inquiry- and estate-types. You may take a look at the currently available resources of the baseline when having a look at the Resources Overview.

These resources and their structure are the same on every FlowFact system, although some inquirytypes for instance may be missing on different systems. If your FlowFact is barely customized you may use the downloadable XSD-Files (which can be found here) in order to build messages for interaction with the API. These files contain information on how messages have to look like in order to be understood by the API. Many parts of the schema files are documented to help you use the schema, for example in a generated object model.

You of course also may GET a resource, take a look at the response, change values and POST/PUT it back to store the changes. This is especially a good idea for becoming familiar with the API and the messaging in general.

The presented resources are, although existent in every FlowFact, only the wireframe of the REST-API and the messages described in the xsd would only apply to a barely customized system. The somewhat tricky part (to some extend) are the messages which are being send to heavily customized FlowFact systems.

MESSAGING - HOW TO COPE WITH DIVERSITY

Because of the diversity of certain FlowFact installations the actual messages being sent back and forth through the API may differ. FlowFact will provide a baseline (described above) with all used resources and the messages needed to interact with those resources, but if you are facing a customized FlowFact chances are, that you might have to adapt/amend the downloadable XSD files which are used to create the needed messages.

In order to find the current layout for your system we managed to create so called blueprint-resources at specific points in the API. In any place the system may differ from one another (for example when describing inquiry types, that were not in the baseline or changes on existing inquiry-types) these resource can be used to request a valid xsd-File which can be used to amend the existing downloaded XSD-Files from this website and to enable you as a developer of a third-party application to comfort the special needs of the customized system. Informations on how you do that can be found in the Special Resources section of this documentation.

After you added the XSD to your "pool of XSD files" you will have a complete understanding (described in XSD) on how the message you are going to send has to look like.

ID and identifier

In the “big” or important resources you will find the two attributes ***id*** and ***identifier***. You will ask yourself, what the difference between these attributes is.
The ***id*** is the unique identifier of a record in a database table. It is a GUID. Usually the user can’t see this id in FlowFact.
The ***identifier*** is a value, which the user can see and even set manually in FlowFact. This ***identifier*** is a string, not a GUID, and not even unique.

Please be careful with using the ***id*** or the ***identifier*** as your reference to these records. The ***identifier*** isn’t unique. The ***id*** is, but only inside of one database. For example, if there is a copy of the database, you can’t differ from which database you get these records.
We recommend using the **URI** of the resource as your reference like it is standard for the REST paradigm.

API Sandbox for Testing

For testing purposes often Developer needs an access to the API without beeing currently customer.

To provide this Test environment the FLOWFACT created the API Sandbox. This Sandbox is a fully Live System of hAPPy with an attached API.

Licensing
Please be aware that if you create a product based on the FLOWFACT API you still need to license the API Usage. For more information and possible ways to this read the article about Licensing.

API Sandbox Data	
API Url	https://flowfactapi.flowfact.com
API Activation Code	apisandbox
API user	apisandbox/klaus.erfolg@immo-erfolg.de
API Password	YOSYGDXU14

Authentication
If you need further instructions on how to access the FLOWFACT API and how to authenticate against it please consider the following a rticle.

Some Example Calls

To get the Data of user "Klaus Erfolg":

https://flowfactapi.flowfact.com/com.flowfact.server/api/rest/v1.0/customers/apisandbox/users/10000200-0000-4012-0018-00001B3B30F6

To get all Activities of user "Klaus Erfolg":

https://flowfactapi.flowfact.com/com.flowfact.server/api/rest/v1.0/customers/apisandbox/users/10000200-0000-4012-0018-00001B3B30F6/activities

To get the company data:

https://flowfactapi.flowfact.com/com.flowfact.server/api/rest/v1.0/customers/apisandbox/company

To get the detail fields ressource:

https://flowfactapi.flowfact.com/com.flowfact.server/api/rest/v1.0/customers/apisandbox/properties

Available Query-Interfaces

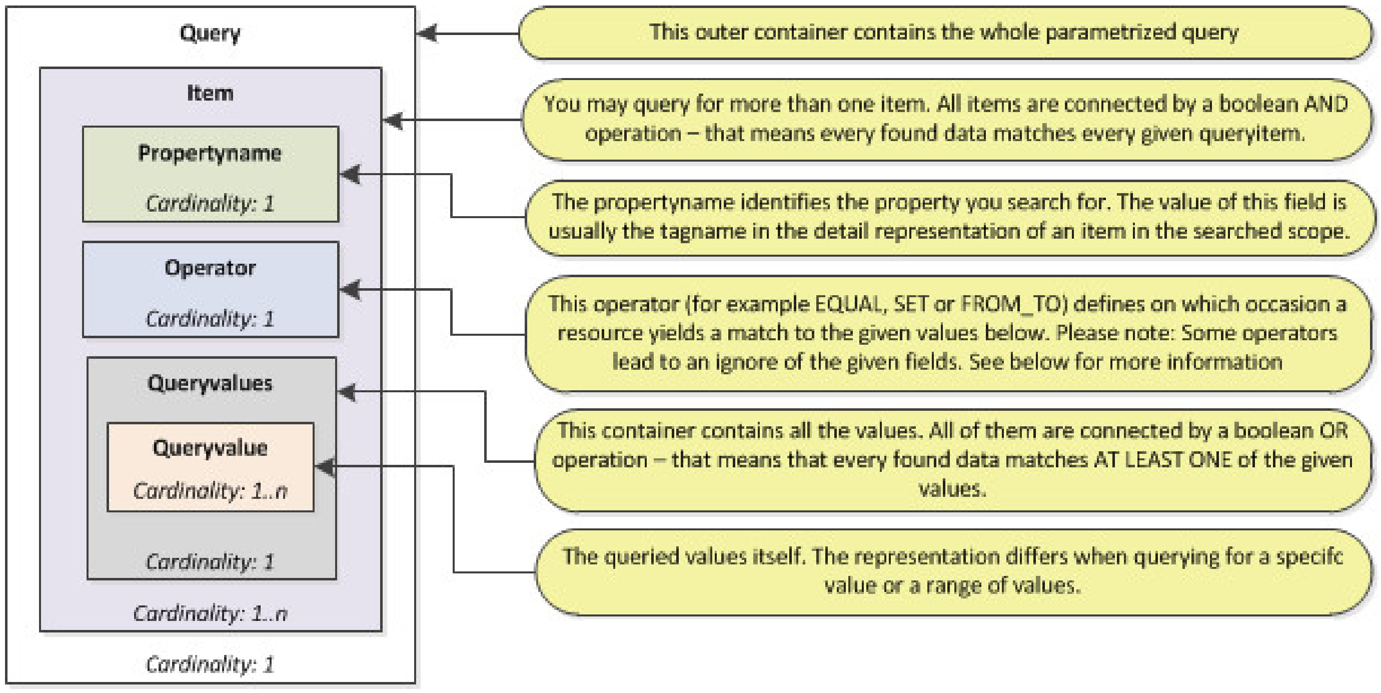
What are query interfaces?

Query interfaces allow you to perform a more detailed search in the context of estates, inquiries, contacts, activities, projects, etc.. Some of the query interfaces are already in place, some of them will be implemented in future releases.

Such an interface is not really REST like, because it can not be bond to a pure CRUD paradigm. However, those interfaces can be used for highly customized searches in order to get a result that is filtered way beyond the simple use of Query-Parameters on list resources.

What is the basic structure of a query?

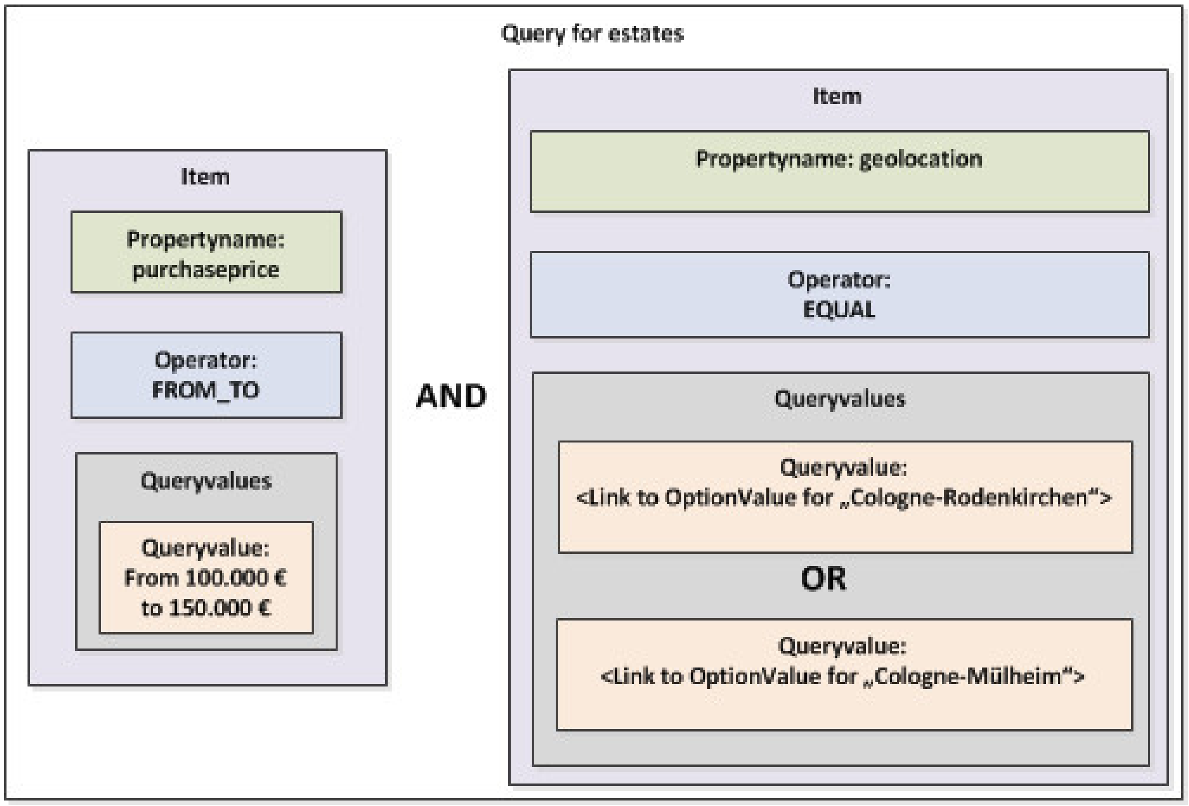
The following diagram shows the basic functionality:


Query
Item
Propertyname
Cardinality: 1
Operator
Cardinality: 1
Queryvalues
Queryvalue
Cardinality: 1..n
Cardinality: 1
Cardinality: 1..n
Cardinality: 1
This outer container contains the whole parametrized query
You may query for more than one item. All items are connected by a boolean AND operation – that means every found data matches every given queryitem.
The propertyname identifies the property you search for. The value of this field is usually the tagname in the detail representation of an item in the searched scope.
This operator (for example EQUAL, SET or FROM_TO) defines on which occasion a resource yields a match to the given values below. Please note: Some operators lead to an ignore of the given fields. See below for more information
This container contains all the values. All of them are connected by a boolean OR operation – that means that every found data matches AT LEAST ONE of the given values.
The queried values itself. The representation differs when querying for a specifc value or a range of values.

One could say you are querying the REST-API like you query a WSDL Webservice: By parametrizing your search into a document which is interpreted by the server you are able to assign any number of parameters to a search. This rather "webservice like" approach dictates the usage of a POST-Method, because every attempt to use resource outside a simple CRUD-paradigm calls for the usage of the not cacheable and non-idempotent (which basically means that any sending of a request will cause a changing on the server, even if the exact same message is sent twice).

As shown in the introduction the basic idea is to encode your query into one or more search items, which are connected by a boolean AND operation. That means you can search for example for estates with a purchaseprice between 100.000 € and 150.000 € AND the geolocationfield of Cologne-Rodenkirchen or Cologne-Mülheim.

As you can see with the example of the geolocation: The actual search values are in a boolean relationship as well: They are connected by boolean OR. The following figure shows the basic outline of a query message with multiple paramteters the query is performed for:


Query for estates
Item
Propertyname: purchaseprice
Operator: FROM_TO
Queryvalues
Queryvalue: From 100.000 € to 150.000 €
AND
Item
Propertyname: geolocation
Operator: EQUAL
Queryvalues
Queryvalue: <Link to OptionValue for „Cologne-Rodenkirchen“>
OR
Queryvalue: <Link to OptionValue for „Cologne-Mülheim“>

How is a specific field addressed when querying?

When you want to perform a query for one or more specific fields of a resource, the field you query for must be contained in the xml tag called propertyname. The tag <propertyname>purchaseprice</propertyname> for instance would query for the purchaseprice, which appears in the detail representation of estates and inquiries.

You may basically search by all the returned tags of the detail representation in the corresponding scope. That also means, that a query for contacts with a purchase price of 100.000 € will most likely not yield any results, because the purchaseprice is not part of the representation of a contact.

How do I query for specific values?

The query item itself contains the query operator and an arbitrary number of queryvalues, which are connected by a boolean OR relation. So, the first thing to do is to select an adequat query operator. The available operators are part of the corresponding XSD. The list below shows the meaning of each available operator:

Operator	Results
EQUAL	A match must be equal to the given value.
FROM_TO	A match must be within the range of the two given values - please see section about querying for interval values below.
GREATER	A match must be a greater value than the given one.
SMALLER	A match must be a smaller value than the given one.
GREATER_EQUAL	A match must at least equal value than the given one.
SMALLER_EQUAL	A match must not exceed the given value (inclusive).
NOT_EQUAL	A match may not be equal to the given value.
NOT_SET	A match may not have the given value set. When using this operator any value will be ignored by the query interface. You may use this operator to query for unset properties.
SET	A match must have the corresponding property set - regardles the actual value. Any given value is being ignored by the server.

STARTS_WITH	A match must start with the given value. This is especially useful for string properties and makes no sense for other values, like boolean values for example.
ENDS_WITH	A match must end with the given value. This is especially useful for string properties and makes no sense for other values, like linked values (characteristics, contacts, etc.) for example.
LIKE	A match contain the given value. This is especially useful for string properties and makes no sense for other values, like boolean values for example.

When you selected an adequat search operator, the second decision is wether it is a non-linked value you are querying for (like a number value or a string value) or the searched value is a linked resource, which is addressable by the used instance of the FlowFact REST-API. An example for these kind of values are characteristics or contacts linked to an estate.

Either way: If the value is not an interval value (please see below for the usage of interval values) to be queried for, it needs to go as a value enclosed field into the list of ORed values. Please see the following snipped of a query message to get an idea for querying one specifc value:

```
>  <item>
>  <propertyname>active</propertyname>
>  <operator>EQUAL</operator>
>  <queryvalues>
>   <queryvalue>
>    <value>
>     <value>true</value>
>    </value>
>   </queryvalue>
>  </queryvalues>
>  </item>
```

As you can see, we parametrized our "query for active resources" to the query item with propertyname "active" (because this is the tagname in a response coming from the server), the search operator EQUAL and the specific boolean value "true".

The querying for linked values is similar to the example above, but you need to put the link into the value field:

```
> <item>
>  <propertyname>characteristic</propertyname>
>  <operator>EQUAL</operator>
>  <queryvalues>
>    <queryvalue>
>      <value>
>     <!-- This is the link to a specific characteristic within this
FlowFact REST-API -->
>
<value>com.flowfact.server/api/rest/v1./customers/FlowFact/characteristi
cs/ED42A4EF-B82C-440A-8FB0-8F12F87BCB45</value>
>      </value>
>    </queryvalue>
>  </queryvalues>
> </item>
```

Why the representation is somewhat stair-like is becoming clear when you continue with the following section regarding the search for interval values.

How do I query for interval or ranged values?

Basically you query for ranged values similar to queries for specific values - at least when it comes to selecting the field and search operator. The followong snippet shows the difference in the representation when querying for a ranged value:

```
> <item>
>   <propertyname>purchaseprice</propertyname>
>   <operator>FROM_TO</operator>
>   <queryvalues>
>     <queryvalue>
>       <valueinterval>
>         <valuefrom>100000</valuefrom>
>         <valueto>150000</valueto>
>       </valueinterval>
>     </queryvalue>
>   </queryvalues>
> </item>
```

As you can see: The two borders for searching are provided within tags called valuefrom and valueto which are again encapsuled in a tag called valueinterval. This query would yield any result within the given borders inclusive the values itself - in this example any objects that have a purchaseprice within 100000 € and 150000 € set.

If you are interested in a more sofisticated example of how to use the query interface please have a look at Tutorial 4.

What happens when I perform a search that could never have any results?

You will receive a HTTP return code of 204: No content. Querying for reasonable data is uppon you as the developer. For example would a query, that contains two items of the same property most likely never yield any result, when the queried values do not match - an estate, for example, can never be situated in Cologne-Rodenkirchen AND Cologne-Müheim.

Is the query interface idiot-proof?

No, as implied in the section above it is unfortunately not.

Due to the possibly very strange configuration of the target system the API cannot decide wether a query is reasonable (or wise for that matter) or not. The query is performed, no matter what. That means you may not expect some error message saying: Hey, the query you are trying to perform does not make sense at all. It lies in the hands of the developer to build correct queries!

Changelog

Connecting & Authentication for API calls

What do you need to connect to the API?

In order to connect to a FLOWFACT API you need the following information:

- ***"Where (in a matter of Internet -URL) is the target API reachable? Is it using https or http?"***
  When you are programming against the FLOWFACT APIs that is hosted by the FLOWFACT company (Products Online Performer CRM or direct hAPPy / Prime) the answer to the questions regarding protocol and server (**Internet-URL**) are answered easily in one line:

https://flowfactapi.flowfact.com
This address is the starting point of your target API.
When operating against a FLOWFACT API that is hosted off-site (FLOWFACT Performer CRM with own installed Tomcat-Server), the only one that could answer your question is the Administrator / Installationpartner of the corresponding FLOWFACT system.

- ***"What are the credentials of a user / a registered App in that system?"***
  in the end of the day you need a **username** and a **password** that is active and has an entry in the BEN-Table of the Target FLOWFACT system. That being said there are multiple ways to get an entry in there: You can use of course a normal user license for example. If the owner of the FLOWFACT system purchased the "full-license" there can be added an "App-User" to this table, that does not consume a license key.

- ***"What is the Customer-ID of the FLOWFACT System installed?"***
  The **customer-id** information can be obtained by the user that is using the flowfact system.

Please note, that the colors used above are used in the next paragraph.

How to authenticate to the REST-API?

Be prepared!
As a way to enforce our licensing mechanism we will switch to an OAUTH2-Based authorization mechanism eventually. **Right now and until further notice the Authentication mechanism below can and should be used**, but it might be a good idea to keep in mind, that someday OAUTH2 will be mandatory when using the FLOWACT API!

In order to successfully communicate with the provided REST-Interface you need a valid authentication. By default every user of your FlowFact System can be allowed to authenticate against the FLOWFACT-API. Please beware, that this has some implications:

- Based on the person, which is authenticated to the API, the returned data and access rights are based on the users access rights configured in the FLOWFACT System.
- Just like in your normal client: Not every person may see or edit all the data! Based on your rights at the system you may edit data while your colleague may not and vice versa.

Keep in mind that some resources are available to all users while other resource are user specific and thus maybe restricted in access. If you try to gain access to one of those resources you will get, as expected, the according HTTP-Error **403: Forbidden**.

BASIC-Auth Authentication

For now the authentication works with the HTTP-Basic-Auth, details can be found in RFC2617. A lot of tools are supporting this kind of authentication mechanism which is why this mechanism is not discussed in detail here. The credentials to use consist of the used customer-ID plus the Username in combination with your password. **The password must have minimum six characters**, otherwise the authentication will fail!

In general the username to login is a combination of customer-ID and username, divided by a "/": **Customer-ID/Username**

And the password is your password 🙂

As an example in curl:

Example User
The following Data are just an example to explain the construction of a rest call. You will not be able to connect against any system with those data because the System and the user do not exist!

If you need a sandbox account please contact the FLOWFACT to license the API usage as described here.

Internet-URL	https://flowfactapi.flowfact.com
Customer-ID	515789
Username	ffapiuser
Password	t0ps3cr3t

```
curl -u 515789/ffapiuser:t0ps3cr3t https://flowfactapi.flowfact.com/com.flowfact.server/api/rest/v1.0/customers/515789/users/ffapiuser
```

This call should result in an overview of your user (User-Resource) if the credentials, server, port and all other settings are correct.

Please keep in mind that a SSL-Encrypted access to the API is highly recommended in order to encrypt the username/password combination!

Conventions and Required Knowledge

Conventions to the documentation

There are a few conventions on this documentation and the REST interface that is described by it. These are the conventions, which apply to this technical documentation:

- Because we want to accomodate a wide variety of customers all over the world, all used text in this documentation (in images as well of course) has to be composed in the english language.
- The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements for the protocols it implements. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST level requirements but not all the SHOULD level requirements for its protocols is said to be "conditionally compliant."
- This document makes excessive use of messages being sent from the client to the server and vice versa. The messages themself alway have the same layout:

```
HTTP-Method <URL>
HTTP-Parameters
HTTP-Body
```

Sometimes for convenience and readability some of the Headers are omitted.
In order to flag outgoing data every line of a **request** begins with the character >. Data being sent from the server to the client (in a **response** for instance) begins with the character <. Example: **Request**:

```
> GET http://example.com/users
> Accept: text/html
```

Response:

```
< HTTP/1.1 200 OK
< Server: HTTPd/1.0
< Date: Sat, 27 Nov 2004 10:18:15 GMT
< Content-Type: application/xml

...
```

Conventions to the documentation

- Every resource MAY contain more than one representation which don't have necessarily be standard (like application/xml or application/json), but derived from one. This feature has reasons in performance.
- Currently there is only a hand full of custom representations - the most used are *-fullyblown or *-overview, for example application/json-overview or application/xml-fullyblown

- Every representation is documented in the corresponding resource

Required Knowledge

What knowledge do I need to understand this documentation?

As you can imagine: This is hard to tell!

This documentation addresses developers for any kind of application or extension that is going to use the FLOWFACT API, a RESTful Webservices. This documentation will **not** cover any basics about REST, RESTful Webservices or any other architectural design pattern that might be part of the technology you want to use. There will be some explanatory notes on important topics to make the API more understandable though.

The beauty of choosing REST over HTTP for the FLOWFACT API is, that you are basically free in the choice of the technology you are using to write your application.

If you are not sure wether your knowledge about RESTful webservices is adequate in order to fully understand the contents of this documentation, try to answer the following questions:

1. What is the basic concept of webservices in general? What does "orchestrating" mean in combination with webservices?
2. What distinguishes REST webservices from WSDL described webservices?
3. Am I firm in using the most common REST implementation, the HTTP protocol? What is the difference between REST and HTTP? Or is it the same?!
4. What HTTP-Methods do exist and what are the conventions when you are using them? What is the difference between a HTTP-PUT and a HTTP-POST call?
5. What do I know about content negotiation when using HTTP?
6. Why is a RESTful interface self descriptive to some extend and how can I "explore" a RESTful web interface?
7. What Tools can I use when developing a client for a restful (HTTP-)webservice?
8. What does HATEOAS mean and what are the key principles when using them?

Many of those questions are discussed in forums or FAQ articles. The authors of this documentation just want to point out, that basic questions would **not** be answered within this documentation.

Useful Weblinks

- How I explained REST to my wife
- Webseite of the REST-Inventor Roy Fielding
- Building a Restful Webservice by example

Useful Tools

- RESTClient for Firefox - Grafic Tool that enables you to call RESTful Webservices and runs within your Firefox web browser

Custom Mappings

Custom Mappings and the FlowFact Baseline - In a nutshell: The idea of the FlowFact Baseline

The **FlowFact Baseline** is a way to cope with the potential clash of interests between a developer, who is developing applications for a preferably high number of target systems and the real estate company which might have a highly customized system. The FlowFact Baseline on the one hand enables the developer to implement applications that are of use to as many customers as possible with little or none customizing of the code. On the other hand the real estate company / real estate broker has access to a wider set of applications that he can use for his system.

The FlowFact Baseline supports this idea by providing a "common interface" to the data representation. When interacting with the API a part of each message may differ from system to system, for example when it comes to setting specific details or using activitytypes in order to create certain kind of activities. By using the Baseline you as a developer can be sure that (if correctly configured) the detail "purchaseprice" always represents the property that identifies the purchase price on the target system, even if the target system is configured with a new property that does not match the initially shipped property field for a purchase price.

What is covered by the Baseline?

As introduced above the Baseline is covering certain areas of a FlowFact system. Currently the following parts are covered:

- 238 properties that are initially shipped with a new FlowFact direct or Performer System have a custom mapping name that is used as a XML tagname. In general this custom mapping name is the normalized english translation of the corresponding property with underscores instead of blank spaces (like for example purchaseprice, headline, renter, etc.). This Custom Mapping Name is a valid value in means of cross-platform calls. in other (easy) words: The property denoted by the XML-Tagname purchaseprice references the purchaseprice of the target system - no matter which definitve property is set as a pruchaseprice in the system.

- 22 Activitytypes
- 140 Estatetypes which are available optionvalues for the option property "estatetype". The estate type is one of the 238 Properties that is part of the property baseline.
- 48 PropertyAssistances

In a future Release there will be a graphical interface where the custom mappings of every system may be viewed and adapted.

How do use parts of the Baseline?

This is pretty easy: When you want to address a CustomMapping for a detail for example you may use the custommappingname as a tagname. When addressing ActivityTypes you supply

```
activities/activitytypes/external_appointment instead of
activities/activitytypes/10000100-0000-0362-0011-00001B3B30F6.
```

The usage is similar for using CustomMappings in origin tags (for PropertyAssistances) or as selected OptionValue in the property estatetype.

Resources Overview API 1.0.35 and upwards

If you experience any trouble with the view you can also load the documentation separatly under:

https://flowfactapi.flowfact.com/com.flowfact.server/web/api-doc/

Special Resources

SystemInfo

The SystemInfo resource is made for debugging purposes. It shows information about the system where the Apache Tomcat is running. This resource has only the plaintext- and the HTML-representation.

Duplicates

With the duplicates resource it's possible to check if there are equal or similar data records. So far there is a duplicate resource only for contacts. Please check Contacts for further information to this specific resource.

Blueprint

The FlowFact CRM Software is a highly customizable system. The users may, according to specific requirements, define which data should be stored in the different entities by adding and or deleting customized fields.

For example a company, which uses The FlowFact CRM Software, wants to send birthday wishes to their customers. So it’s necessary to store the birthday date of the customer. The FlowFact CRM software offers the possibility to define, that customers should have the property “birthday” with a valid date set to it.

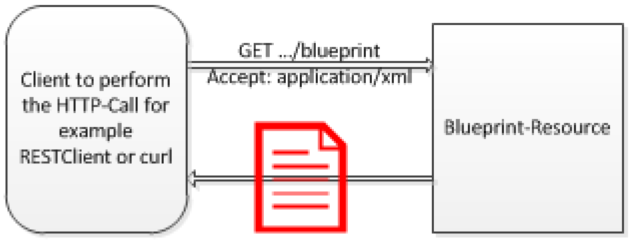
Many users of FlowFact CRM Software have many self-defined properties. This is why it is not possible to create one universal xml-schema for all resources, because the representation may very well differ from system to system.

To solve this problem the blueprint resources were made. They deliver an xml-schema (XSD) of these customized properties. So if a customized property should be used, the developer has to fetch these schemas from the coresponding resources' blueprint resource and place them to the other xml-schemas in order to generate the entity classes. The following figure should clarify the process:

Step 1: Download available XSD from http://restapi.flowfact.de

Downloaded XSD Files

Step 2: You encounter a customized resource, lets say the inquiry-type for „appartments for rent" has another field that you want to read/write. Don't worry – just read out the current configuration of your system and build your message according to it!


Client to perform the HTTP-Call for example RESTClient or curl
GET .../blueprint
Accept: application/xml
Blueprint-Resource

Step 3: Add the generated XSD file to your set of XSD files. If there are changes to an existing type you would have to overwrite the corresponding schema file. In this case it might be a good idea to store a backup of that file.

New set of xsd-Files

Step 4: Take look at the new XSD, figure out how the customized fields are described and which types are available. Adapt your client and you are good to go!

Please note that if the user changes his properties (add new properties, changes properties …) these changes are not known by any implemented client and will be ignored until the developer gets the current xml-schema from the blueprint resource and handle these changes in the client software.

Currently blueprint resources are available to inquiry-types and contactdetails. They are planned for activities and estates as well.

Initialize

Resource is used to trigger the initialization of all property values by setting a nicename for every value to the data store. You do not need to call this resource, because missing nicenames will be added automatically during runtime. Calling this resource using a POST command will save some resources during runtime though.

Portals

With FlowFact it's possible to transmit estates to estate online portals like Immobilienscout24.de. This resource gets a list of the configurations to these portals.
You can find this resource here:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/portals
> Accept: application/xml
```

If you want to send an estate to one of these portals via FlowFact, you will need this resource.

Placeholders

FlowFact offers a function to replace specific placeholders in texts like brochure texts. These placeholders can stand for information like a contact-name or address, or even for pictures.
These defined placeholders are reachable in this resource.
You will need them, when you try to upload a picture to an estate.

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/placeholders
> Accept: application/xml
```

Tutorials & Best Practices

This section is about Tutorials and Best Practices on performing special usecases when using the FLOWFACT API.

- Best Practice: Paging Data from the FLOWFACT API
- Tutorial 1: Interacting with contacts
- Tutorial 2: Activities and Inquiries
- Tutorial 3: Estate attachments
- Tutorial 4: Query-Interface

Best Practice: Paging Data from the FLOWFACT API

INTRODUCTION

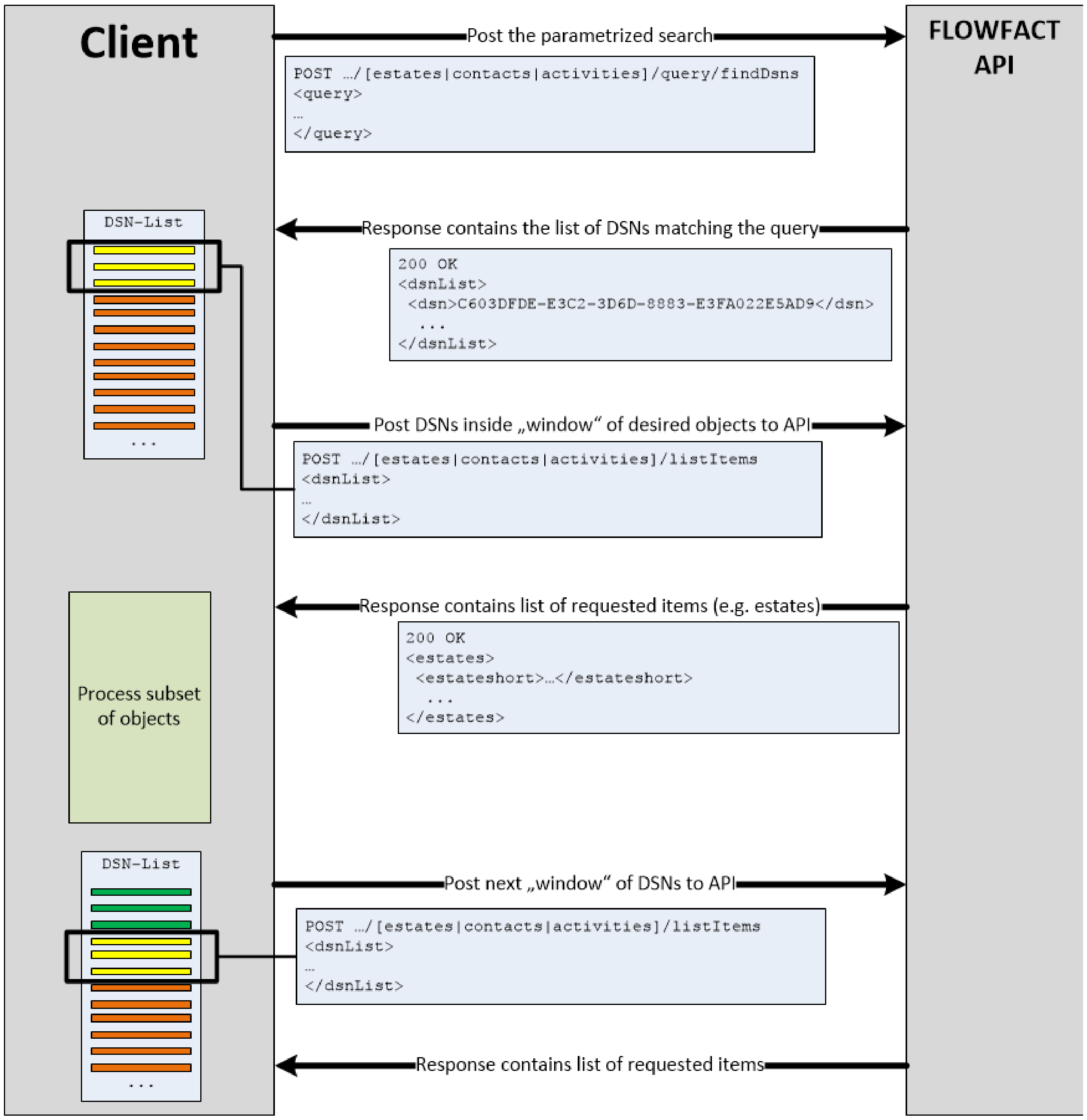
Paging data is in constant request by many developers, that are working against the FLOWFACT API. Many of them are struggling with big databases and a high latency of responses to their Queries when using the paging mechanism that is currently provided. Unfortunately the FLOWFACT system can be run on older MSSQL database systems, which either do not support paging operations at all or quite poorly. In order to still being able to operate with large datasets, the FLOWFACT API offers a specific mechanism to do so.

This mechanism was introduced with Version 1.0.14.6 of the FLOWFACT API and is described in the following in detail.

WHEN SHOULD I USE THIS APPROACH?

Whenever you want to read a potentially big amount of estates, contacts or addresses from the API you could use this approach. It is especially useful if you want to do an initial fetch of data after which you just load modified data, for example when using it with some kind of website development that presents for example active estates of a real estate agent.

BASIC WORKFLOW


Client
FLOWFACT API
Post the parametrized search
Response contains the list of DSNs matching the query
DSN-List
Post DSNs inside „window“ of desired objects to API
Response contains list of requested items (e.g. estates)
Process subset of objects
DSN-List
Post next „window“ of DSNs to API
Response contains list of requested items

```
POST …/[estates|contacts|activities]/query/findDsns
<query>
…
</query>
```

```
200 OK
<dsnList>
 <dsn>C603DFDE-E3C2-3D6D-8883-E3FA022E5AD9</dsn>
  ...
</dsnList>
```

```
POST …/[estates|contacts|activities]/listItems
<dsnList>
…
</dsnList>
```

```
200 OK
<estates>
 <estateshort>…</estateshort>
  ...
</estates>
```

```
POST …/[estates|contacts|activities]/listItems
<dsnList>
…
</dsnList>
```

● ● ●

STEP 1: BUILDING QUERYTYPE AND SEND IT TO THE API

First step is to build a Query that can also be used for the Query-Resource. In this example we are querying all estates that

- have the characteristic with id 22F2D2AF-F762-3775-A4F6-9B8820AB9CD9 attached to it
- are not archived

The result is ordered by active state and the creation date descending (last modification first in response). The corresponding request looks like the following:

Send QueryType to API

```
POST
http(s)://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers/[c
ustomerid]/users/[username|userid]/estates/query/findDsns

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<query xmlns="http://www.flowfact.com/schema/rest">
 <item>
  <propertyname>CHARACTERISTIC</propertyname>
  <operator>EQUAL</operator>
  <queryvalues>
   <queryvalue>
    <value>
     <value>22F2D2AF-F762-3775-A4F6-9B8820AB9CD9</value>
    </value>
   </queryvalue>
  </queryvalues>
 </item>
 <item>
  <propertyname>ARCHIVED</propertyname>
  <operator>EQUAL</operator>
  <queryvalues>
   <queryvalue>
    <value>
     <value>false</value>
    </value>
   </queryvalue>
  </queryvalues>
 </item>
 <order>
  <name>ACTIVE</name>
  <asc>true</asc>
 </order>
 <order>
  <name>CREATED</name>
  <asc>false</asc>
 </order>
</query>
```

This request is being send to the API which evaluates the request and calculates the DSNs of the objects, that are applying to the query. A response looks like the following:

Response from findDsns

```
200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dsnList xmlns="http://www.flowfact.com/schema/rest">
  <dsn>74603729-E694-32F2-B929-74E13CDFEE60</dsn>
  <dsn>13B675EF-8CF1-31E5-BDB5-7933555103E5</dsn>
  <dsn>811E6076-673D-3725-963E-8858724A0603</dsn>
  <dsn>7E75CEB3-560B-3D88-886E-C4B03C0B62F2</dsn>
  <dsn>AF1C2803-5A67-3C54-81B4-DE9E4608C837</dsn>
  <dsn>EE3E4652-7AB2-382D-84C9-FA0AA4047D83</dsn>
  <dsn>745F975E-0361-3B8D-B187-39CF780075E6</dsn>
  <dsn>4AF9051A-E6BC-3696-8C73-8EEFA03737D6</dsn>
  <dsn>D974BB1D-0964-36DE-80A0-70CBBD79A9A5</dsn>
  <dsn>93B82DE2-481D-3AEB-B2D9-B26DB69F929B</dsn>
  <dsn>EB2E1FEB-5492-3657-9329-79DBA4AB08B0</dsn>
  <dsn>5F7437BE-700D-30B9-B5D6-6A148FBAA02F</dsn>
  <dsn>F40814F4-F3D7-3345-8875-A57060A1C39B</dsn>
  <dsn>F05CD605-D88B-3F7B-A859-D13F76973A58</dsn>
  <dsn>65EA04C2-F432-34AF-9942-0F5670776A50</dsn>
  <dsn>F82A1715-4D07-36EB-AC1C-E4BFE51B1229</dsn>
</dsnList>
```

Please take note, that this list can be very (!) long (depending on the query you sent). However, the calculation is quite fast - in our tests we were getting 3138 results of a total of 3265 estates with the query above in 217ms.

STEP 2: TRUNCATE REPLIED DATA INTO CONVINIENT SLICES AND GET RESULTS FROM THE API

Once we obtained the list above we can go ahead an truncate this list the way it suits our purpose. For the sake of this tutorial we will truncate the list above in two parts with 10 entries each. So our next call looks like the following:

Response from findDsns

```
POST
http(s)://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers/[c
ustomerid]/users/[username|userid]/estates/listItems

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dsnList xmlns="http://www.flowfact.com/schema/rest">
  <dsn>74603729-E694-32F2-B929-74E13CDFEE60</dsn>
  <dsn>13B675EF-8CF1-31E5-BDB5-7933555103E5</dsn>
  <dsn>811E6076-673D-3725-963E-8858724A0603</dsn>
  <dsn>7E75CEB3-560B-3D88-886E-C4B03C0B62F2</dsn>
  <dsn>AF1C2803-5A67-3C54-81B4-DE9E4608C837</dsn>
  <dsn>EE3E4652-7AB2-382D-84C9-FA0AA4047D83</dsn>
  <dsn>745F975E-0361-3B8D-B187-39CF780075E6</dsn>
  <dsn>4AF9051A-E6BC-3696-8C73-8EEFA03737D6</dsn>
  <dsn>D974BB1D-0964-36DE-80A0-70CBBD79A9A5</dsn>
</dsnList>
```

The request is being calculated and replied. This is also a quite fast operation - calculation in the scenario above took 110ms. The reply looks like the following:

Response from findDsns

```
200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<estates xmlns="http://www.flowfact.com/schema/rest">
  <total>0</total>
  <estateshort identifier="8266"
id="745F975E-0361-3B8D-B187-39CF780075E6" rel="8266 - 1429837464992"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/745F975E-0361-3B8D-B187-39CF78007
5E6">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>1429837464992</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8264"
id="D974BB1D-0964-36DE-80A0-70CBBD79A9A5" rel="8264 - MWA59
ESTATE_1429836785875"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/D974BB1D-0964-36DE-80A0-70CBBD79A
```

```
9A5">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA59 ESTATE_1429836785875</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8272"
id="74603729-E694-32F2-B929-74E13CDFEE60" rel="8272 -
MWA142_1429862546622"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/74603729-E694-32F2-B929-74E13CDFE
E60">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA142_1429862546622</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8271"
id="13B675EF-8CF1-31E5-BDB5-7933555103E5" rel="8271 -
MWA142_1429860427236"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/13B675EF-8CF1-31E5-BDB5-793355510
3E5">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA142_1429860427236</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8270"
id="811E6076-673D-3725-963E-8858724A0603" rel="8270 -
MWA147_1429838909457"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/811E6076-673D-3725-963E-8858724A0
603">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA147_1429838909457</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8265"
id="4AF9051A-E6BC-3696-8C73-8EEFA03737D6" rel="8265 - 2222"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/4AF9051A-E6BC-3696-8C73-8EEFA0373
7D6">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>2222</internaldescription>
    <headline>2222</headline>
    <location>
      <postalcode>2222 2222</postalcode>
    </location>
  </estateshort>
  <estateshort identifier="8269"
id="7E75CEB3-560B-3D88-886E-C4B03C0B62F2" rel="8269 - MWA74
ESTATE_1429838564683"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/7E75CEB3-560B-3D88-886E-C4B03C0B6
2F2">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA74 ESTATE_1429838564683</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8268"
id="AF1C2803-5A67-3C54-81B4-DE9E4608C837" rel="8268 - MWA73
ESTATE_1429838034002"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/AF1C2803-5A67-3C54-81B4-DE9E4608C
837">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA73 ESTATE_1429838034002</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8267"
id="EE3E4652-7AB2-382D-84C9-FA0AA4047D83" rel="8267 - MWA72
ESTATE_1429837797868"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/EE3E4652-7AB2-382D-84C9-FA0AA4047
D83">
    <archived>false</archived>
```

```
<active>true</active>
<objecttracking>false</objecttracking>
<internaldescription>
</internaldescription>
<headline>MWA72 ESTATE_1429837797868</headline>
```

```
    <location/>
  </estateshort>
</estates>
```

Now the response can be evaluated and processed on client side.

STEP 3: FETCH FOR MORE DATA

Once processing the data is finished the next (and in this example last) slice of data can be fetched from the server:

Response from findDsns

```
POST
http(s)://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers/[c
ustomerid]/users/[username|userid]/estates/listItems

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<dsnList xmlns="http://www.flowfact.com/schema/rest">
  <dsn>93B82DE2-481D-3AEB-B2D9-B26DB69F929B</dsn>
  <dsn>EB2E1FEB-5492-3657-9329-79DBA4AB08B0</dsn>
  <dsn>5F7437BE-700D-30B9-B5D6-6A148FBAA02F</dsn>
  <dsn>F40814F4-F3D7-3345-8875-A57060A1C39B</dsn>
  <dsn>F05CD605-D88B-3F7B-A859-D13F76973A58</dsn>
  <dsn>65EA04C2-F432-34AF-9942-0F5670776A50</dsn>
  <dsn>F82A1715-4D07-36EB-AC1C-E4BFE51B1229</dsn>
</dsnList>
```

...and the response...

Response from findDsns

```
200 OK

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<estates xmlns="http://www.flowfact.com/schema/rest">
  <total>0</total>
  <estateshort identifier="8257"
id="65EA04C2-F432-34AF-9942-0F5670776A50" rel="8257 - MWA48
ESTATE_1429832671953"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/65EA04C2-F432-34AF-9942-0F5670776
A50">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA48 ESTATE_1429832671953</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8261"
id="5F7437BE-700D-30B9-B5D6-6A148FBAA02F" rel="8261 - MWA57-1 ESTATE"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/5F7437BE-700D-30B9-B5D6-6A148FBAA
02F">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA57-1 ESTATE</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8262"
id="EB2E1FEB-5492-3657-9329-79DBA4AB08B0" rel="8262 - MWA57-2 ESTATE"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/EB2E1FEB-5492-3657-9329-79DBA4AB0
8B0">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA57-2 ESTATE</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8260"
id="F40814F4-F3D7-3345-8875-A57060A1C39B" rel="8260 - MWA56-1
ESTATE_1429834585325"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/F40814F4-F3D7-3345-8875-A57060A1C
39B">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA56-1 ESTATE_1429834585325</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8263"
id="93B82DE2-481D-3AEB-B2D9-B26DB69F929B" rel="8263 - MWA58
ESTATE_1429836593535"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/93B82DE2-481D-3AEB-B2D9-B26DB69F9
29B">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA58 ESTATE_1429836593535</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8259"
id="F05CD605-D88B-3F7B-A859-D13F76973A58" rel="8259 - MWA54
ESTATE_1429833049150"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/F05CD605-D88B-3F7B-A859-D13F76973
A58">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA54 ESTATE_1429833049150</headline>
    <location/>
  </estateshort>
  <estateshort identifier="8256"
id="F82A1715-4D07-36EB-AC1C-E4BFE51B1229" rel="8256 -
MWA158_ESTATE_1429831971791"
href="http://[server]:[port]/com.flowfact.server/api/rest/v1.0/customers
/[customerid]/users/[username]/estates/F82A1715-4D07-36EB-AC1C-E4BFE51B1
229">
    <archived>false</archived>
    <active>true</active>
    <objecttracking>false</objecttracking>
    <internaldescription>
    </internaldescription>
    <headline>MWA158_ESTATE_1429831971791</headline>
    <location/>
  </estateshort>
</estates>
```

Please take note, that it is of course not needed to request the findDsns Resource everytime you are requesting another window of data! It is suggested to store the list of dsns temporary and work with it during the process.

Tutorial 1: Interacting with contacts

This tutorial talks you through a basic process of receiving data from the service, changing it and create new data at the server by the example of contacts. In fact in this tutorial not even the whole feature set of contacts are being mentioned. This is why contact details and the attaching/detaching of characteristics will be covered in another tutorial. The purpose of this tutorial is to get used to the REST-API and the recommended patterns when interacting with the interface. Let's start!

Contacts in general are user scoped, which means that a user called Alice MAY invoke operations on user Bob's contacts, but maybe not likewise. In order to cope with this kind of user based content restriction contacts and some other resources are categorized by using the user's DSN in the url. The server MUST only return the data which is visible by the given user. When dealing with the ffApiUser the User-DSN would be FADFCEF3-95BD-42D6-9A35-0F07770041C1 by default.

A little advise before we start: The used examples in this tutorial were written during the development of the XML Schema which means, that the shown examples may not match 100% the current version of the API. Please always refer to the newest version of the xsd and resource layout to implement your workflow and build your messages - the aim of the following tutorial is to point out examples for interacting with the service, not the absolutely correct usage!

Enough said, lets get down to business and start slow: In general this interface provides CRUD-functionality to contacts. Contacts are not only adresses - they are more than that. They consist of different names, keywords, salutation phonenumbers, banking information, attached characteristics or adresses and, last but not least, contactdetails that may or may not be attached to them (which are unfortunately system specific - this issue is covered in Tutorial 2!). The following sections will cover CRUD-Operations - we will start with receiving contacts.

HOW DO I RECEIVE MY CONTACTS?

As pointed out before contacts are user scoped. We are interested in a xml representation of the data. So a possible request for the list-ressource contacts could be:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts
> Accept: application/xml
```

If we would have fired the call above without any valid authentication we would almost instantly get a *401: Authorization Required* response:

```
< HTTP/1.1 401 Authorization Required
< Server: HTTPd/1.0
< Date: Sat, 27 Nov 2004 10:18:15 GMT
< WWW-Authenticate: Basic realm="FF REST-API"
< Content-Type: application/xml
< Content-Length: 169

< <error>
<   <message>You are not authenticated to this server!</message>
<   <suggestions>
<    <suggestion>Please supply a valid authentication!</suggestion>
<   </suggestions>
< </error>
```

What happened? We are not logged in yet (or forgot to include our basic auth token in the request parameter...) and this is the servers authentication challenge. So the client receives this response and provides a valid base64 encoded username/passwort combination. The server MAY reply with a session id which is only valid for a short period of time. The server MUST use appropriate and standardized mechanisms to do so!

If you are using the FireFox RESTClient plugin in order to testdrive the REST-API you can enter your credentials in a seperate window. Please keep in mind, that the login name consists of you FlowFact-Customer-ID (or Databasename when using a performer), followed by a slash and you FlowFact username with which you would login to the client.

Anyway, in this example we are using plain old basic auth (security issues are imminent which is why every server SHOULD use a SSL encrypted connection) and the correct request would look something like this:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts

> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0
```

```
< HTTP/1.1 200 OK
< Content-Type: application/xml
< Content-Length: 576

< <contacts>
<  <contactshort
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/878005
39-c16b-43a2-96ed-6c62452c79a3">
< <name>Alice Anderson</name>
< <street>An der Wachsfabrik 23</street>
< <postalcode>50999</postalcode>
< <city>Cologne</city>
<  </contactshort>
<  <contactshort
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/566c15
c6-3a07-472b-bdac-643836c58730">
< <name>Bob Bugallow</name>
< <street>Industriestr. 161A</street>
< <company>FlowFact AG</company>
< <postalcode>50999</postalcode>
< <city>Cologne</city>
<  </contactshort>
<  <contactshort
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/13e1c4
6a-418d-473d-9d55-251d2d9ccc3e">
< <name>Carol Cross</name>
< <street>Evergreen Terrace 123</street>
< <company>ACME Corp.</company>
< <postalcode>10161</postalcode>
< <city>New York, NY</city>
< <country>United States</country>
<  </contactshort>
< </contacts>
```

Please note, that we did not add any additional Accept-Header, which would are denote the desired representation for the response message. This means we will receive the **standard representation of this list-resource**, which is basically a **short overview** of the items in that list. A resource MAY provide more than one representation and contacts is one of those resources, which provides at least four representations:

```
application/xml, application/json: standard (short) representation for
list resources
application/full+xml, application/full+json: fully blown xml object
representations
```

There could be more or less than the representations above for other resources, like an additional text/html or just a text/plain representation, but this depends on the resource and the implementation. The important thing is, that the **client is able to get a fully blown or an overview representation in different representation formats**.

So, if we wanted to receive a set of fully blown contacts the request for people in our database that live in the area with zip code 50999 would look something like this:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts?postalcode="
50999"

> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0
> Accept: application/full+xml
```

```
< HTTP/1.1 200 OK
< Content-Type: application/full+xml
< Content-Length: ...

< <contacts>
< <?xml version="1.0" encoding="UTF-8"?>
< <contact
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/878005
39-c16b-43a2-96ed-6c62452c79a3" type="application/xml,application/json"
      xmlns="http://www.flowfact.com/schema/rest"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<   <readonly>true<readonly>
<   <identifier>12345<identifier>
<   <archived>false</archived>
<   <name>
<     <lastname>Anderson</lastname>
<     <firstname>Alice</firstname>
<   </name>
<   <keyword1>Colleague</keyword1>
<   <salutation>Mrs.</salutation>
<   <location>
<     <street>An der Wachsfabrik 23</street>
<     <city>Cologne</city>
<     <postalcode>50999</postalcode>
<     <additionalAdressInformations>
<       <additionalAdressInformation>Appartment
2A</additionalAdressInformation>
<     </additionalAdressInformations>
<     <district>Cologne-Sürth</district>
<   </location>
<   <phonenumbers>
<     <office>
<       <countrycode>44</countrycode>
<       <areacode>15478</areacode>
<       <subscribernumber>546215</subscribernumber>
<       <note>Office number - call only from nine to five!</note>
<     </office>
<   </phonenumbers>
<   <emailaddresses>
<     <primaryemail>alice@anderson.de</primaryemail>
<   </emailaddresses>
<   <webpage>http://tempuri.org</webpage>
<   <note>Nice colleague, fair boss!</note>
<   <characteristics>
<     <characteristic
href="https://example.com/rest/contracts/5156789/characteristics/d5c970e
8-3696-4f16-b98a-599cd62af016" rel="Boss"/>
<     <characteristic
href="https://example.com/rest/contracts/5156789/characteristics/9ffee96
d-a700-48a6-b8e6-b88793bb7c75" rel="House Owner"/>
<   </characteristics>
< </contact>
< <contact
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/878005
39-c16b-43a2-96ed-6c62452c79a4" type="application/xml,application/json"
      xmlns="http://www.flowfact.com/schema/rest"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<   <readonly>false<readonly>
<   <identifier>6789<identifier>
<   <archived>false</archived>
<   <company>FlowFact AG</company>
<   <name>
<     <lastname>Bugallow</lastname>
<     <firstname>Bob</firstname>
<   </name>
<   <salutation>Mr.</salutation>
<   <location>
<     <street>Industriestr. 161</street>
<     <city>Cologne</city>
<     <postalcode>50999</postalcode>
<     <district>Cologne-Rodenkirchen</district>
<   </location>
<   <emailaddresses>
<     <primaryemail>bob@ffag.de</primaryemail>
<   </emailaddresses>
<   <webpage>http://flowfact.de</webpage>
<   <note>Does not like coffee.</note>
<   <characteristics>
<     <characteristic
href="https://example.com/rest/contracts/5156789/characteristics/22d97a7
9-46d5-478a-9248-65742976878c" rel="Bob Bugallow"/>
```

```
<   </characteristics>
< </contact>
< </contacts>
```

As you can see the received information by the client is much more detailed than in the short representation, because we are dealing with a lot more data. This may take a while when requesting, for example, all the contacts in the database (in some cases this would mean to read and parse more than 250.000 Addresses!). In order to cope with the requirement for detailed data and the scalability of the presented information the REST-Interface implements a way to limit the transferred data by a set of simple query parameters. By default every service implementation of a list resource which represents potentially much data (like contacts) MUST implement the query parameters **size** and **page**. By using these two parameters a simple paging mechanism is being implemented to navigate through the data and avoid unnecessary load to the server. If you are interested in all the possible query-parameters please check out the contacts-section of this documentation.

HOW DO I RECEIVE A SINGLE CONTACT?

You basically receive a single contact the same way you received a list of contacts - plus the information about the specific contact you want to read out, stated as a DSN:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/87800539-c16
b-43a2-96ed-6c62452c79a3

> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0
```

The URLs of the available contacts are denoted as the href-Attribute in every short representation of a contact. This pattern (linkage by href-values) is adapted from HTML, where Media and links are connected as href-values as well. So, if you decide to take a closer look (read out the full representation of a contact which is listed), you just use the given href-value to get to it. The answer in this case would be something like:

```
< HTTP/1.1 200 OK
< Content-Type: application/full+xml
< Content-Length: ...

< <contacts>
< <?xml version="1.0" encoding="UTF-8"?>
< <contact
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/878005
39-c16b-43a2-96ed-6c62452c79a3" type="application/xml,application/json"
      xmlns="http://www.flowfact.com/schema/rest"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<   <readonly>true<readonly>
<   <identifier>12345<identifier>
<   <archived>false</archived>
<   <name>
<     <lastname>Anderson</lastname>
<     <firstname>Alice</firstname>
<   </name>
<   <keyword1>Colleague</keyword1>
<   <salutation>Mrs.</salutation>
<   <location>
<     <street>An der Wachsfabrik 23</street>
<     <city>Cologne</city>
<     <postalcode>50999</postalcode>
<     <additionalAdressInformations>
<       <additionalAdressInformation>Appartment
2A</additionalAdressInformation>
<     </additionalAdressInformations>
<     <district>Cologne-Sürth</district>
<   </location>
<   <phonenumbers>
<     <office>
<       <countrycode>49</countrycode>
<       <areacode>221</areacode>
<       <subscribernumber>8800</subscribernumber>
<       <note>Office number</note>
<     </office>
<   </phonenumbers>
<   <emailaddresses>
<     <primaryemail>alice@anderson.de</primaryemail>
<   </emailaddresses>
<   <webpage>http://tempuri.org</webpage>
<   <note>Nice colleague, fair boss!</note>
<   <characteristics>
<     <characteristic
href="https://example.com/rest/contracts/5156789/characteristics/d5c970e
8-3696-4f16-b98a-599cd62af016" rel="Boss"/>
<     <characteristic
href="https://example.com/rest/contracts/5156789/characteristics/9ffee96
```

```
d-a700-48a6-b8e6-b88793bb7c75" rel="House Owner"/>
<   </characteristics>
< </contact>
```

The most obvious difference from requesting a set of contacts is, that the request for a single contact contains an identifier in the called URL, which distinctly identifies this contact. As you can see there is another difference in the request between receiving a list of contacts and receiving a single contact: The default representation!

When you are dealing with entity resources such as one specific contact, the standard representation is application/xml or application/json and represents always the **fully blown** object. Scalability is no real issue here, because we are only dealing with a single entity - in this case a contact. Additionally an entity resource MAY also provide an overview representation. This behaviour is working for JSON as well.

HOW DO I CREATE A NEW CONTACT?

In fact you already know how to create a new contact when you know your way around RESTful webservices: You use the HTTP-Command **POS T** on the corresponding list-resource (contacts in this case) in order to invoke the creation of a new contact. The request-response pair for creating a bare contact (only the field "name" is set) may look like this:

```
> POST
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts

> Content-Type: application/xml
> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0

> <?xml version="1.0" encoding="UTF-8"?>
> <contact href="" xmlns:p="http://www.flowfact.com/schema/rest"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
>   <name>
>     <lastname>Dumbledore</lastname>
>   </name>
> </contact>
```

```
< HTTP/1.1 201 Created
< Location:
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/c0d32bb2-a30
5-4e8d-a08a-d10de5e10e27
< Content-Type: application/xml

< <?xml version="1.0" encoding="UTF-8"?>
< <contact
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/c0d32b
b2-a305-4e8d-a08a-d10de5e10e27"
       xmlns:p="http://www.flowfact.com/schema/rest"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<   <readonly>false</readonly>
<   <identifier>45689</identifier>
<   <name>
<     <lastname>Dumbledore</lastname>
<   </name>
< </contact>
```

Of course you are allowed to create a more detailed contact, but keep in mind that some fields SHOULD NOT be set during creation or editing of some entity types. When dealing with contacts those fields are, for example, the identifier or the readonly attribute, which MUST be ignored by the server when received in an PUT or POST message. In general one could say, that attribute values SHOULD NOT be altered by the client.

When the contact could be created succesfully the return code of this request MUST be *201: Created* followed by the URL in the Location-Header which points to the newly created entity. The server MAY include the generated entity when responding to the request as shown in the example above.

HOW DO I UPDATE AN EXISTING CONTACT?

Imagine Mrs. Anderson married and decided to change her lastname to the double barrel name "Anderson-Erhard". Additionally after her marriage she moved into a new flat with her husband. In general invoking changes on already existing entities are invoked by HTTP-PUT messages to the entitys' URL. Before we changed the data, we requested the entity of that specific contact. So the request- and response-message for changing the entity would look something like this:

```
> POST
https://yourserver.com/api/rest/customers/5156789/alice/contacts/8780053
9-c16b-43a2-96ed-6c62452c79a3

> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0
> Content-Type: application/xml
> <?xml version="1.0" encoding="UTF-8"?>
> <contact
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/878005
39-c16b-43a2-96ed-6c62452c79a3" type="application/xml,application/json"
      xmlns="http://www.flowfact.com/schema/rest"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
>   <readonly>true<readonly>
>   <identifier>12345<identifier>
>   <archived>false</archived>
>   <name>
>     <lastname>Anderson-Erhard</lastname>
>     <firstname>Alice</firstname>
>   </name>
>   <keyword1>Colleague</keyword1>
>   <salutation>Mrs.</salutation>
>   <location>
>     <street>Ubierring 157</street>
>     <city>Cologne</city>
>     <postalcode>50678</postalcode>
>   </location>
>   <phonenumbers>
<     <office>
<       <countrycode>44</countrycode>
<       <areacode>15478</areacode>
<       <subscribernumber>546215</subscribernumber>
<       <note>Office number - call only from nine to five!</note>
<     </office>
>   </phonenumbers>
>   <emailaddresses>
>     <primaryemail>alice@anderson.de</primaryemail>
>   </emailaddresses>
>   <webpage>http://tempuri.org</webpage>
>   <note>Nice colleague, fair boss!</note>
>   <characteristics>
>     <characteristic
href="https://example.com/rest/contracts/5156789/characteristics/d5c970e
8-3696-4f16-b98a-599cd62af016" rel="Boss"/>
>     <characteristic
href="https://example.com/rest/contracts/5156789/characteristics/9ffee96
d-a700-48a6-b8e6-b88793bb7c75" rel="House Owner"/>
>   </characteristics>
> </contact>
```

```
< HTTP/1.1 200 OK
```

Thats it. Another GET request to the entity returns the last PUTted information. Some entities may have additional fields like a last edited timestamp that are updated by the server uppon a PUT request.

Of course there may be errors when the client tries to update data - for example would

```
<emailaddresses>
   <primaryemail>NotCorrectlyFormattedEmail</primaryemail>
</emailaddresses>
```

yield an HTTP-Error **400 Bad request**, because it does not fit the predefined format for mail addresses denoted in the xsd. These Bad request errors also appear if an error is being made by the client that could not be penpointed to a specific misbehaviour. The body of the returned message MAY contain information about what went wrong.

Another error the client MUST expect would be a HTTP-Error **403 Forbidden** which is being returned for example if you are trying to edit a read-only ressoure. Please keep in mind, that a 403 generally means that you are trying to do something you are not allowed to do - providing a valid authentication (in your current context) would not solve the issue!

HOW DO I DELETE A CONTACT?

This is by far the easiest and most dangerous operation if you are not absolutely sure what you are doing. It is invoked by a HTTP DELETE command. The request and reply would be like:

```
> DELETE
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/87800539-c16
b-43a2-96ed-6c62452c79a3

> Authorization: Basic YWxpY2U6dDBwczNjcjN0
```

```
< HTTP/1.1 200 OK
```

As you can see: There is no doublechecking like "Are you really sure that you want to delete this contact?" of any kind - if you invoke the DELETE command, it is performed (if you are authorized to do so of course).

If you are not aware of this fact, a delete action is quite bad for entity-objects - but unintentially fired it is even worse (lets say: a worst-case scenario) for list-objects. Imagine: The RESTful pattern says, that if you invoke a DELETE command on a list resource **all of the underlying resources are being deleted**. This is what the service implementation MUST do (if the method DELETE is provided): Deleting all underlying resources under the corresponding list resource. Deleting an entity does not necessarily mean that the data is really deleted, but it MUST NOT appear as available resource to the client using the REST interface.

Tutorial 2: Activities and Inquiries

IMPLEMENTING SPECIFIC WORKFLOW USING THE REST-API

In Tutorial 1 you learned how to deal with the basic process of receiving data from the service and pushing data to it by examplary calls in the contacts' context. In this tutorial you learn how to implement a whole workflow for an exemplary website, which uses certain kind of dynamic calls to the REST-API to implement specific workflows. Please keep in mind: This is only an **example** of a possible workflow - your workflow may of course differ!

SCENARIO

Imagine you are searching for a nice house in a specific area to buy. Your research on the internet revealed a website of a local estate broker which allows you to search for estates this broker offers. A first search brought some results, but although the offered estates are interesting they do not fully match your requirements. However, the websites provides some kind of registration. Registered customers are enabled to stay in close contact with the broker. Wouldn't it be sweet, if you could send your requirements for your new house to the website of the estate broker and get an email or receive a phone call when a matching estate is available?

Now imagine you are the estate broker. In your world, as often, Time is money, so you try to work more efficiently. But every time you get a visitor (or message or call or email,...) from a person interested in an estate, you have to store his data and requirements manually using your current installation of FlowFact. Many customers with active and up-to-date inquiries are very important for your work, but also very time-consuming to manage. Wouldn't it be awesome, if exisitng and new customers are able to do this work for you?

We have good news for the both you as a web-designer! All the above use cases and many more are possible with this REST-API (and a corresponding website implementation). The REST-API is like a toolbox: It depends on the estate broker's business workflows and your creativity to create an interactive website, which brings advantages to all involved parties.

In this Tutorial the following steps will be covered:

- Registration and authorization of a new user to the website. This user is added to the FlowFact system using the REST-API.
- The user may add/alter his contact information after logging in as an authorized user.
- A new inquiry may be added to the FlowFact system by using the REST-API. Old inquiries can be listed and set as "inactive" when they do not apply anymore.
- The user may send his broker (or one of his employees) a short information about an estate he wants to sell or lease. This information, with attached files, is sent directly to one of the broker's employees by creating an activity.

REGISTRATION AND AUTHENTICATION OF THE END USER TO THE WEBSITE

As a website designer you would have to manage the registration of customers to the website, because you would not enable the customer of the estate broker to use the REST-API directly. You have to build a registration process, with which you can enable/disable specific users of the website to do certain things, like supplying inquiries or updating their contact informations. There are many ways to cope with the requirement of a registration and this documentation does not dig in, but it basically boils down to some kind of datastore (i.e. database) in which the users of the website are stored alongside their rights to use the API funtionality.

The first step in using the REST-API would be to create this registration process in which you shield the REST-API from unauthorized usage.

Take the following table fields as an example of the storage of users credentials:

column	description
email	The email-address of the interested person (in this example this is also the login for the user to the website)
password	The encrypted password for the login
FF_DSN	The ID of the related contact in FlowFact CRM (description will follow in this tutorial)

Please keep in mind that none of the stored passwords have anything to do with data, that is stored in your FlowFact-Backend! The stored usernames and passwords are simply a matter of authorizing the user to your website.

That being said, your table of registered users to your website may look like this:

email	password	FF_DSN
hugo.kern@trash-mail.com	Fl0wFactRul3z (encrypted)	C1F40C86-CE28-4734-8413-838AC6FC25CD
abc@yahoo.de	Test42 (encrypted)	69A1E0CE-F97F-4FA8-9063-5760D6820A4C

What the actual process of authorizing a customer looks like depends on your policy. A possible way to authorize new users to your website may be by requiring a visit to the brokers office by the user who is interested in using the website for his/her inquiries. Another possibility would be to authorize users by their credit card information or maybe your policy does not require any authorization at all and every visitor to the website may use the features of the dynamic website. However your process looks like, in the end of the customized authentication process registered users should be stored in a datastorage of some kind that is accessible by your website. The important information you would need for further interaction is the users contact DSN - this DSN describes the corresponding DSN of the contact in the FlowFact system. How to fill in the correct DSN for the user is depending heavily on your process used for authentication.

In this tutorial we follow the following steps to authenticate a user:

- The user registers himself at the website using a form which asks for basic data. The submit of the form is guarded by a Captcha-Mechanism.
- A new contact is created by using the REST-API with the given information. The created user DSN is stored from the response on the creation request. The creation and alteration of contact information is described in Tutorial 1 ("How do I create a new contact")

- A duplicate search is being performed. When duplicates turn up an activity is being created where the created contact is being linked as address1 and the possible duplicates are stored as AddressLinks in the detail view of that activity (please see "Resource Duplicates" in contacts for further information)
  If no duplicates turn up just an activity is being created telling an employee that there has been another registration to the website - this helps the estate brokers backoffice to keep track of new registrations.
- When the contact was a duplicate the DSN in the websites database has to be changed - the login information stays the same and the user won't notice a thing! After this happened either automatically or manually by an employee the rights of the new user (wether he may or may not see new inquiries, etc.) are being set.

CREATING ACTIVITIES FOR CONTACTS

In our workflow after creating a contact it is necessary to create an activity. Either we create an activity, which is used to cope with duplicates or, in this case, we just create an activity which tells the backoffice "Hey, there has been a new registration to the website!".

In this case we need a simple activity, which is basically consists of a message with a subject, a body and some other information. What information the activity should contain is up to you and the estate broker - whatever fits the needs is possible. In this example we would use the activitytype "API" to notify uppon a new registered user of the website. We are supplying some information, like date, duedate, reminder and the link to the created contact to simply keep track of the created contact. After creation we will attach a characteristic to the activity.

The following call creates a simple activity where the newly created contact is connected to:

```
> POST
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities

> Content-Type: application/xml
> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0

> <?xml version="1.0" encoding="UTF-8"?>
> <activity href="" xmlns:p="https://www.flowfact.com/schema/rest"
xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
>   <origin rel="API"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/propertyassistances/activitytypes/D653D995-CB51-4F6C-ACBF-
531F0454896A"/>
>   <address1 rel="Hugo Kern"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/C1F40C
86-CE28-4734-8413-838AC6FC25CD"/>
>   <date>2012-07-29T16:25:00.000+02:00</date>
>   <duedate>2012-07-30T16:25:00.000+02:00</duedate>
>   <alarm>2012-07-30T15:25:00.000+02:00</alarm>
>   <archived>false</archived>
>   <done>false</done>
>   <priority>0</priority>
>   <subject>New Contact was created by the REST-API</subject>
>   <body>Please verify the contact.</body>
> </activity>
```

As you can see there is an "address1"-link in the example. This is the link to the new contact, consisting of the url to the contact in the system (baseurl plus the DSN that identifies this exact contact). The DSN has to be available in your datastorage at this point.

The other important link is the link "origin". This is the link to the specification of this activiytype. With this link we specify this activity as an activity of type "API". The header and the body contains information about the required steps, that should be (in our scenario) done manually by the estate broker or one of his employees. If the creation was successful, you will receive a positive response:

```
< HTTP/1.1 201 Created
< Location:
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/12CA6B23-1
903-3182-B5AF-EEFFC963609F
< Content-Type: application/xml

< <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
< <activity readonly="false" identifier=" 2386708" rel="Test API Sun Jul
29 16:24:12 CEST 2012"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/12CA
6B23-1903-3182-B5AF-EEFFC963609F">
<   <origin rel="API"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/propertyassistances/activitytypes/D653D995-CB51-4F6C-ACBF-
531F0454896A"/>
<   <owner rel="ffApiUser"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1"/>
<   <address1 rel="Hugo Kern"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/C1F40C
86-CE28-4734-8413-838AC6FC25CD"/>
<   <colors>
<     <fgcolor>#0</fgcolor>
<     <bgcolor>#FFFFFF</bgcolor>
<   </colors>
<   <accesslist>
<     <everybodyaccess>
<       <right>FULL</right>
<     </everybodyaccess>
<   </accesslist>
<   <creation>2012-08-23T15:43:26.086+02:00</creation>
<   <duedate>2012-07-26T16:40:17.737+02:00</duedate>
<   <reminder>2012-07-26T16:40:17.737+02:00</reminder>
<   <date>2012-07-26T16:40:17.737+02:00</date>
<   <subject>New Contact was created by the REST-API</subject>
<   <body>Please verify the contact.</body>
<   <archived>false</archived>
<   <done>false</done>
<   <priority>0</priority>
< </activity>
```

As you can see: Some of the informations of the activity (like "colors" or "creation") are being set automatically to the created activity.

Now that we created the activity it is time to attach some characteristics to it, in order to make it appear in a users todo-list.

ATTACH CHARACTERISTICS TO AN ACTIVITY

An activity (and many other resources) may have attached characteristics. These characteristics may represent persons (estate broker, employees,...) or other logical units described in your FlowFact system. Let us attach the characteristic of the estate broker to the new activity, so he should find this activity in his to-do list.

```
> POST
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/12CA6B23-1
903-3182-B5AF-EEFFC963609F/characteristics

> Content-Type: application/xml
> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0

> <?xml version="1.0" encoding="UTF-8"?>
> <characteristics xmlns="https://www.flowfact.com/schema/rest">
>   <characteristic
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/characteristics/BA097B90-5C61-46E1-85F0-7405BE9881E7"/>
> </characteristics>
```

If attaching the characteristic was successful, you will receive a positive response containing the list of currently added characteristics to this activity:

```
< HTTP/1.1 200 OK
< Content-Type: application/xml

< <?xml version="1.0" encoding="UTF-8"?>
< <characteristics>
<   <characteristic rel="Karl Zufrieden"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/characteristics/BA097B90-5C61-46E1-85F0-7405BE9881E7">
<     <chartype>RESOURCE</chartype>
<     <colors>
<       <fgcolor>#0</fgcolor>
<       <bgcolor>#FFFFFF</bgcolor>
<     </colors>
<     <name>Karl Zufrieden</name>
<     <active>true</active>
<     <shortname>zuf</shortname>
<     <grouppath>>Mitarbeiter</grouppath>
<   </characteristic>
< </characteristics>
```

Lets doublecheck the whole activity. As you can see the attached characteristic is supplied by the overview of the activity:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/12CA6B23-1
903-3182-B5AF-EEFFC963609F

> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0

< 200 OK
< <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
< <activity readonly="false" identifier=" 2386708" rel="Test API Sun Jul
29 16:24:12 CEST 2012"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/12CA
6B23-1903-3182-B5AF-EEFFC963609F">
<   <origin rel="API"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/propertyassistances/activitytypes/D653D995-CB51-4F6C-ACBF-
531F0454896A"/>
<   <owner rel="ffApiUser"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1"/>
```

```
<   <address1 rel="Hugo Kern"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/C1F40C
86-CE28-4734-8413-838AC6FC25CD"/>
<   <characteristics
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/C2E2
1269-CF0A-3719-8110-FA600437D21D/characteristics">
<     <characteristic rel="Karl Zufrieden"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/characteristics/BA097B90-5C61-46E1-85F0-7405BE9881E7"/>
<   </characteristics>
<   <colors>
<     <fgcolor>#0</fgcolor>
<     <bgcolor>#FFFFFF</bgcolor>
<   </colors>
<   <accesslist>
<     <everybodyaccess>
<       <right>FULL</right>
<     </everybodyaccess>
<   </accesslist>
<   <creation>2012-08-23T15:43:26.086+02:00</creation>
<   <duedate>2012-07-26T16:40:17.737+02:00</duedate>
<   <reminder>2012-07-26T16:40:17.737+02:00</reminder>
<   <date>2012-07-26T16:40:17.737+02:00</date>
<   <subject>New Contact was created by the REST-API</subject>
<   <body>Please verify the contact.</body>
<   <archived>false</archived>
<   <done>false</done>
<   <priority>0</priority>
< </activity>
```

CREATING AN INQUIRY

Once the interested person is stored as a contact in FlowFact Backend system and the contact is validated by the estate broker, the interested person may login to the website. At the beginning we talked about a possibility to store search-requirements for an estate. These requirements are called "inquiries" in a FlowFact system. How do we cope with different kind of inquiries?

Imagine three people. One of them wants to buy a nice house for his family, the other needs a flat for himself and the last looks for a new office. You can help this three (and many more) people with inquiries. But wait, they search completely different types of estates. Doesn't that mean, that someone, who looks for a house for his family has other requirements on an estate, than someone who is looking for a flat or an office? Yes, thats right. Therefore inquiries are also categorized into estate types, much like actitivity types. Every inquiry-type offers other possibilities to store requirements to a search for estates. So if you are looking for a house, you will probably get the possibility to enter your requirement for the size of the land area or the amount of floors the house has got.

If you’re looking for a flat the land area size doesn't matter in most cases. So the inquiry-type for flats probably won’t contain this possibility.

Those inquiry-types have to be identified by the "origin" tag of the inquiry. Similar to activity-types the administrator of a FlowFact Performer CRM system can define own inquiry-types for his purposes, so they have to by dynamic to fit all needs.

That being said: Let’s create an inquiry for a flat:

```
> POST
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/inquiries

> Content-Type: application/xml
> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0

< 201 Created
< Location:
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/propertyassistances/inquirydetails/5B6FE8F1-B106-4312-BB66-421E2
ECC3718

< <?xml version="1.0" encoding="UTF-8"?>
< <inquiry>
<   <origin  rel="Wohnungen"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/propertyassistances/inquirydetails/5B6FE8F1-B106-4312-BB66
-421E2ECC3718"/>
<   <archived>false</archived>
<   <active>true</active>
<   <contact
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/31B1B6
95-104E-4947-8EBE-5DE8C7A0C2D8"/>
<   <details>
<     <kaufpreis name="Kaufpreis">
<       <ns2:musthave>true</ns2:musthave>
<       <ns2:rating>GOOD</ns2:rating>
<       <ns2:valuefrom>
<         <ns2:value>80000.0</ns2:value>
<         <ns2:unit>€</ns2:unit>
<       </ns2:valuefrom>
<       <ns2:valueto>
<         <ns2:value>80000.0</ns2:value>
<         <ns2:unit>€</ns2:unit>
<       </ns2:valueto>
<     </kaufpreis>
<     <ObjArt name="Objektart">
<       <ns2:musthave>true</ns2:musthave>
<       <ns2:rating>NORMAL</ns2:rating>
<       <ns2:selected rel="Wohnungen"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/properties/option/A829CAD6-44FF-43C5-9C08-4118E49F4D73/ava
ilableoptions/80A12695-CFD0-4229-998F-B9355F4B4B57"/>
<       <ns2:options rel="Available Options..."
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/properties/option/A829CAD6-44FF-43C5-9C08-4118E49F4D73/ava
ilableoptions"/>
```

```
<     </ObjArt>
<   </details>
< </inquiry>
```

As you see, there are some infomations in an inquiry, which every inquiry contains. The dynamic part of the inquiry, which depends on the inquiry-type, is the part "details". The possible tags in this part can and probably will differ from each other in the diverse inquiry-types or FlowFact CRM systems.
As we created this API, we couldn’t know, which details our customers have, so it wasn’t possible to generate xsd-files for the messages. For this purpose we created the "blueprint" resource. You can checkout in the inquiry-part of this documentation and see how it works.

This inquiry has got two details, that are required by matching objects: The purchaseprice and the estatetype. The given rating-values have impact on r a later selection done by your Performer or direct client. A selection itself is not yet supported by the REST-API in v

The inquiry you just created is visible to any user in the FlowFact system which is allowed to see it. In this case there is no access restriction.

CREATING A NEW ACTIVITY AND ADD ATTACHMENTS

The last part of this tutorial covers the ability of the REST-API to attach files to an activity. The usecase would be, for example, to allow registered users to supply information about an estate they want to sell/lease. The API will support adding estates directly, but in this scenario an employee of the estate broker will add a new estate via his FlowFact system based on the information conveyed by the user.

The general steps to create a new activity is similar to the process described above, which is why we will skip those steps. Just consider an activity of type "API" to be created. Requesting a representation for this activity may look like this:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/29EAC17D-8
E51-4E81-9308-2F5493649CB3

> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0

< 200 OK
< <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
< <activity readonly="false" identifier=" 2386708" rel="Test API Sun Jul
29 16:24:12 CEST 2012"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/29EA
C17D-8E51-4E81-9308-2F5493649CB3">
<   <origin rel="API"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/propertyassistances/activitytypes/D653D995-CB51-4F6C-ACBF-
531F0454896A"/>
<   <owner rel="ffApiUser"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1"/>
<   <address1 rel="Hugo Kern"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/C1F40C
86-CE28-4734-8413-838AC6FC25CD"/>
<   <characteristics
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
```

```
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/C2E2
1269-CF0A-3719-8110-FA600437D21D/characteristics">
<     <characteristic rel="Karl Zufrieden"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/5156789/characteristics/BA097B90-5C61-46E1-85F0-7405BE9881E7"/>
<   </characteristics>
<   <colors>
<     <fgcolor>#0</fgcolor>
<     <bgcolor>#FFFFFF</bgcolor>
<   </colors>
<   <accesslist>
<     <everybodyaccess>
<       <right>FULL</right>
<     </everybodyaccess>
<   </accesslist>
<   <creation>2012-08-23T15:43:26.086+02:00</creation>
<   <duedate>2012-07-26T16:40:17.737+02:00</duedate>
<   <reminder>2012-07-26T16:40:17.737+02:00</reminder>
<   <date>2012-07-26T16:40:17.737+02:00</date>
<   <subject>Mr. Kern wants to have an estate traded by you!</subject>
<   <body>The user wants to trade his estate! He supplied the following
information:
<   Good day to you sir!
<   Here is the information about my estate I want to sell. Please let
me know wether you are up to the job! It is a House which has a
floorsize of 150m²
<   and a plot size of 400m². Please also see the attached pictures! I
would be happy if you could find a buyer for this estate :)
<   Greetings
<   Kern
<   </body>
<   <archived>false</archived>
<   <done>false</done>
<   <priority>0</priority>
<   <attachments rel="Attachments"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
```

```
omers/5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/29EA
C17D-8E51-4E81-9308-2F5493649CB3/attachments"/>
< </activity>
```

As you can see the body contains static information ("The user wants to trade...") and information, which may be supplied by a textfield on the website which was filled out by the user. What exactly is written to the body is up to the webdeveloper and should aid the employee to process the customers wish.

In the response, which represents the basic foundation of our activity, also the link to the attachments is present. In order to add files to this activity we are going to use this link! First we check out which files are currently attached to the activity:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/29EAC17D-8
E51-4E81-9308-2F5493649CB3/attachments

> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0

< 200 OK
< <attachments/>
```

As you can see: Right now there are no attached files present! Our website allows the user to add one photo, which allows a first glance of the estate he wants to sell. This picture may be added to the activity:

```
> POST
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/29EAC17D-
8E51-4E81-9308-2F5493649CB3/attachments?filename=Overview.png&send=true
> Content-Type: application/octet-stream

> <...binary data...>

< 200 OK
<   <attachment rel="Overview.png"
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/29E
AC17D-8E51-4E81-9308-2F5493649CB3/attachments/5B6ADEA1-AAFB-48B7-99EB-64
192F47CD95">
<       <send>true</send>
<       <name>Overview.png</name>
<       <size>255458</size>
<       <file
href="https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/cust
omers/FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/activities/29E
AC17D-8E51-4E81-9308-2F5493649CB3/attachments/5B6ADEA1-AAFB-48B7-99EB-64
192F47CD95/file"/>
<   </attachment>
```

As you can see: Not only the attachment has been uplodaded but also metadata has been created for the new attachment. How to further handle (edit, detach, etc.) attachments to activities please see the section about activities in this documentation.

Tutorial 3: Estate attachments

ATTACHING FILES TO AN ESTATE

In this tutorial you will learn how to attach pictures and file attachments to an existing estate. This workflow comes in handy when 3rd party applications (like a website or a Mobile App) should add estates to the portfolio of estates and attach pictures and floorplans with it.

SCENARIO

In this Tutorial we are facing a PHP website, that is using the REST-API in order to add estates and pictures/files to the estate. The creation of an estate is not covered in detail, because the workflow is analogue to creating a contact, inquiry, etc.. Just use a POST command to estates and supply the necessary data (please see the corresponding part in the API documentation).

READ OUT THE EXISTING ESTATE

To present a basic overview to the used estate we first request a representation of the currently set data to the estate. The request and its (partly) response looks like this:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39A
-499C-B05E-6142F6BC6111
```

```
> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0


< 200 OK
< <estate identifier=" 37230" id="274D495A-C39A-499C-B05E-6142F6BC6111"
rel=" 37230 - House to buy" href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39
A-499C-B05E-6142F6BC6111" >
<  <archived>false</archived>
<  <active>false</active>
<  <objecttracking>false</objecttracking>
<  <created>2012-07-02T11:20:35.000+02:00</created>
<  <modified>2012-07-02T11:20:35.000+02:00</modified>
<  <origin rel="1-2 Familienhaus_Kauf" href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/propertyassistances/estatetypes/D6CAA0BD-617D-4E87-8D83-6188AAC
A5490" />
<  <internaldescription>H-Bad Bergzabern,</internaldescription>
<  <keyword/>
<  <vendor rel="Mr. Miller" href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/8EB40682-A8
96-46F3-8E8C-CC116188F897" />
<  <source rel="Mr. Miller" href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/contacts/8EB40682-A8
96-46F3-8E8C-CC116188F897" />
<  <tradetype>PURCHASE</tradetype>
<  <foreignidentifier/>
<  <children rel="Children of this estate" href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates?parent=http:
//yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/FlowFa
ct/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39A-499C
-B05E-6142F6BC6111" />
<  <pictures rel="Pictures" href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39
A-499C-B05E-6142F6BC6111/pictures" >
<   <previewimage href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39
A-499C-B05E-6142F6BC6111/pictures/main/file" />
<   <total>0</total>
<  </pictures>
<  <attachements rel="Attachments" href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39
```

```
A-499C-B05E-6142F6BC6111/estateattachments" >
<   <total>0</total>
<  </attachements>
<  <characteristics href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39
A-499C-B05E-6142F6BC6111/characteristics" >
<   <characteristic rel="Rhinearea" href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39
A-499C-B05E-6142F6BC6111/characteristics/9C257B3F-7675-494C-8D0A-54D4534
F6223" />
<  <characteristic rel="Jack Salesman" href=
"http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39
A-499C-B05E-6142F6BC6111/characteristics/0D58558D-C5BB-4CF2-B63C-54B1915
482DA" />
```

```
<  </characteristics>
<  ...
< </estate>
```

Most of the presented data above is not relevant for this tutorial. In fact the only two tags that are of importance are the attachments- and pictures-tags. These two tags are of type link, which means their href-attribute points to another resource, and contain some additional data of attached files/pictures. In this case there are neither files nor pictures attached, which is why the value in the total-Tag is zero.

ATTACH IMAGES TO THE ESTATE

A very important part when dealing with estates are the corresponding pictures. The REST-API allows you to up- and download pictures to/from an estate. Every picture that is attached to an estate has a corresponding **placeholder**. A placeholder is like a label, that puts the picture in a specific role in the context of the estate. The main picture placeholder for example signalizes a portal that this picture should be choosen as preview image while a picture that has an internal placeholder assigned is not transferred to the portal at all. The important thing to keep in mind is, that you have to supply a placeholder in order to upload a picture to the estate.

Usage of placeholders

Now you may ask which placeholders are available in your system. You can call them by the following request:

```
> GET
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
FlowFact/placeholders

> Accept: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0


< 200 OK
< <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
< <placeholders xmlns="http://www.flowfact.com/schema/rest">
<  <placeholder id="63B49BBD-DDF8-4F29-9A9F-014E5EA4CEA0" rel="doc2"
href="http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/custo
mers/FlowFact/placeholders/63B49BBD-DDF8-4F29-9A9F-014E5EA4CEA0">
<   <name>doc2</name>
<   <online>true</online>
<   <sorting>2</sorting>
<   <created>2012-04-12+02:00</created>
<   <modified>2012-04-12+02:00</modified>
<  </placeholder>
<  <placeholder id="E2D3F277-DAD9-41D2-9B44-088B3A93B9FC"
rel="Hauptbild"
href="http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/custo
mers/FlowFact/placeholders/E2D3F277-DAD9-41D2-9B44-088B3A93B9FC">
<   <name>Hauptbild</name>
<   <online>true</online>
<   <sorting>1</sorting>
<   <created>2012-01-03+01:00</created>
<   <modified>2012-01-03+01:00</modified>
<  </placeholder>
<  <placeholder id="1B6E58D1-11F7-9716-B1A8-16F819F88D84" rel="Intern9"
href="http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/custo
mers/FlowFact/placeholders/1B6E58D1-11F7-9716-B1A8-16F819F88D84">
<   <name>Intern9</name>
<   <online>false</online>
<   <sorting>108</sorting>
<   <created>2012-01-03+01:00</created>
<   <modified>2012-01-03+01:00</modified>
<  </placeholder>
<   ...
< </placeholders>
```

As you can see each of the placeholder entries has a href value pointing to a specific placeholder. This is the value you need in order to attach a picture (or a file) to an estate.

Uploading an image

Im order to upload an image we have to use the picture-subresource of a specific estate. In the example above this would be http://yourserver.co m:8080/com.flowfact.server/api/rest/v1.0/customers/FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39A-499C-B05E-6142F6BC6111/pictures. In order to upload a picture you have to issue a POST Request onto this resource which looks like this:

```
> POST
http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39A-499C-B05E-6142F6BC6111/pictures?filename=Overview.png&placeholder=com.flowfact.server/api/rest/v1.0/customers/FlowFact/placeholders/E2D3F277-DAD9-41D2-9B44-088B3A93B9FC

> Accept: application/xml
> Content-Type: multipart/form-data
> Authorization: Basic YWxpY2U6dDBwczNjcjN0
```

Please see, that there are two query parameters used at the called URL, which are both mandatory in order to upload an image. The parameter **filename** is the filename of the uploaded picture and needed in order to calculate the correct MIME-Type and for later use in you Performer or Direct Client. In addition to a filename you have to provide a link to a placeholder that is available in your system. This is done by adding the **placeholder** query parameter, which contains the link to a specific placeholder.

Unfortunately the functionality of uploading pictures or files in general is not testable with the RESTClient Tool of Firefox, which is suggested to use for debugging purposes when interacting with the REST-API. This is why we present to you a small PHP-Code Snippet, which uploads an Image to an estate in order to make the process of uploading more understandable.

```
<?php
 error_reporting(E_ALL);

 if(isset($_GET['action']) AND $_GET['action'] == "1") {
  //The actual uploading is done here - please see the the set headers
and other data

  move_uploaded_file($_FILES['datei']['tmp_name'],
$_FILES['datei']['name']);
  $file = "C:\\xampp\htdocs\\";
  $file .= $_FILES['datei']['name'];

  $url =
'http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39A-499C-B05E-6142F6BC6111/pictures?filename=Overview.png&placeholder=com.flowfact.server/api/rest/v1.0/customers/FlowFact/placeholders/E2D3F277-DAD9-41D2-9B44-088B3A93B9FC';
  $username = "FlowFact/ffApiUser";
  $password = "flowfact";

  $ch = curl_init($url);

  curl_setopt($ch, CURLOPT_HTTPAUTH, constant('CURLAUTH_' .
strtoupper('any')));
  curl_setopt($ch, CURLOPT_USERPWD, $username . ':' . $password);

  // Include header in result? (0 = yes, 1 = no)
  curl_setopt($ch, CURLOPT_HEADER, 0);

  curl_setopt($ch, CURLOPT_HTTPHEADER, array("Content-Type:
multipart/form-data"));

  // Should cURL return or print out the data? (true = return, false =
print)
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

  // Timeout in seconds
  curl_setopt($ch, CURLOPT_TIMEOUT, 10);

  curl_setopt($ch, CURLOPT_POST, true);
  $post = array(
   "userfile" => "@$file"
  );

  curl_setopt($ch, CURLOPT_POSTFIELDS, $post);
  $response = curl_exec($ch);
  echo $response;

  curl_close($ch);
 } else {
  //Display simple upload page - not much more than an inputfield and a
button
  echo "<a href='?action=1'>API</a>";
  echo "<br /><br />";
  echo "<form action='apitest.php?action=1' method='post'
enctype='multipart/form-data'  accept-charset='UTF-8'>";
  echo " <input type='file' name='datei'>
     <br>
    <input type='submit' value='Upload'>";
  echo "</form>";
 }
?>
```

The code above presents a simple upload page when called. You may pick a file from your system and hit the "Upload" button. The file is uploaded to the PHP-Server which calls the REST-API at once (triggered by a query param passed to the url after clicking "Upload"). Please see, that the filename is fix in this PHP Snippet, but it could also be calculated from the selected file or be chosen in a different way.

The answer, in this case, would look like this:

```
< 200 OK
< <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
< <estatepicture xmlns="http://www.flowfact.com/schema/rest"
id="5F9F9356-1646-3C14-A210-7ECA0BAE5516" rel="Overview"
href="http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/custo
mers/FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/B8EEBC6
D-44C5-4A13-8945-58E675A4F732/pictures/5F9F9356-1646-3C14-A210-7ECA0BAE5
516">
<  <filesize>22324</filesize>
<  <placeholder rel="Hauptbild"
href="http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/custo
mers/FlowFact/placeholders/E2D3F277-DAD9-41D2-9B44-088B3A93B9FC"/>
<  <created>2012-12-14+01:00</created>
<  <modified>2012-12-14+01:00</modified>
<  <file rel="file"
href="http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/custo
mers/FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/B8EEBC6
D-44C5-4A13-8945-58E675A4F732/pictures/5F9F9356-1646-3C14-A210-7ECA0BAE5
516/file"/>
<  <headline>Overview</headline>
< </estatepicture>
```

This information is added to the estate overview as well. If you call the estate again you will now see, that there is one estate picture and its meta-data. By calling the URL linked at the file-tags href-attribute in a normal Browser-Window (RESTClient wouldn't work here either) the binary file is being downloaded.

As you might have noticed: We used a special placeholder to upload the file to - the placeholder for the main picture. The configuration of every REST-API instance contains one placeholder that is used as the placeholder for main pictures. The picture at this placeholder can be referenced by calling the URL http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39A-499C-B05E-6142F6BC6111/pictures/main/file, which would directly lead you to the preview image of this estate, regardless the used main placeholder of the system.

For both, the estates pictures and attachments the following rules apply when uploading binary data:

- If you choose to upload a file with a placeholder, that is already set, the existing image / attachment for the placeholder will be overwritten and cannot be recovered
- If you just want to change the headline of an attached file or picture and/or switch a placeholder between two already uploaded files/pictures, you may use a POST method to do so. It would look like this:

```
> POST
http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/custom
ers/FlowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274
D495A-C39A-499C-B05E-6142F6BC6111/pictures/5F9F9356-1646-3C14-A210-
7ECA0BAE5516

> Accept: application/xml
> Content-Type:  application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0

> <estatepicture xmlns="http://www.flowfact.com/schema/rest">
>  <placeholder
href="http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/
customers/FlowFact/placeholders/1B6E58D1-11F7-9716-B1A8-16F819F88D8
4"/>
>  <headline>New Overview</headline>
> </estatepicture>
```

After that operation the new placeholder for this file is "intern9", as shown in the list of placeholders above and the headline would be "New Overview". Please note, that you are just changing metadata here - no binary data is exchanged!

- If you want to switch a file to a placeholder that is already in use, you receive a 400: Bad Request that tells you which picture is connected to the desired placeholder. However, you can tell the API to switch the placeholders by adding the query parameter **?switchPlaceHoldersFlag=true**. This query parameter tells the API that you are okay with the fact, that your call has effects on other resources than the one you are currently addressing.

- You may delete an attached file/picture by issuing a DELETE message on the URL value of a specific item in the list.

ATTACH FILES TO THE ESTATE

Attaching a file is being performed analogous to attaching a picture. In this case however you have to pick a document placeholder. An appropriate call would look like this:

```
> POST
http://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/F
lowFact/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/estates/274D495A-C39A
-499C-B05E-6142F6BC6111/estateattachments?filename=Overview.png&com.flow
fact.server/api/rest/v1.0/customers/FlowFact/placeholders/63B49BBD-DDF8-
4F29-9A9F-014E5EA4CEA0

> Accept: application/xml
> Content-Type: multipart/form-data
> Authorization: Basic YWxpY2U6dDBwczNjcjN0
```

Tutorial 4: Query-Interface

USING THE QUERY-INTERFACE (QI) TO SEARCH FOR ESTATES AND INQUIRIES

By using query parameters presented to you in each section of this documentation you already learned about the possibility to simply filter results for your request. For example when searching for activities you may add characteristics and a remindertime to your GET request in order to retrieve a todo list.

However, FlowFact is a CRM System with much more functionality than just displaying data for a handful of searchable values. One might feel the urge to query the system for more complex data in order to perform a special task with all found entities. In order to cope with this situation, the REST-API provides a resource called Query-Interface. This interface is able to perform queries based on the given parameters, which are most of the fields that are displayed to you when you receive a response message.

How the query interface is structurized can be seen in the corresponding chapter in the API documentation. Here we want to create a specific query and evaluate the response.

SCENARIO

Imagine a real estate agent is "out in the field" and is trying to pitch an estate he is supposed to sell. There is more than one real estate agent lusting for the oppurtunity so sell the estate and an agent needs a really good reason to get the job. What would be better than querying his FlowFact System with all its data in order to present the owner a list of at least 30 active inquiries that more or less fit the owners house parameters and target price.

Or lets turn the tables! Again, imagine a real estate agent, that is working on a sunny saturday evening in a pedestrian area in order to show interested people that walk by what he has to offer. The people tell him what they are willing to pay, where they want to live and some more parameters and bazinga - the agent can present them the results based on live data at his iPad or Android Tablet.

In this tutorial we would dig into the first scenario and see how the request/response pattern would look like using the query interface.

QUERYING FOR INQUIRIES

In general every main scope of the REST-API (estates, inquiries, contacts and activities) has or will have its own query interface. Those interface resources can be found directly beneath the corresponding list resource by adding the token /query to the URL. In this example we are using the query interface of the inquiries resource.

```
> POST
https://yourserver.com:8080/com.flowfact.server/api/rest/v1.0/customers/
5156789/users/FADFCEF3-95BD-42D6-9A35-0F07770041C1/inquiries/query

> Accept: application/xml
> Content-Type: application/xml
> Authorization: Basic YWxpY2U6dDBwczNjcjN0

> <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
> <query xmlns="http://www.flowfact.com/schema/rest">
>     <item>
>   <propertyname>purchaseprice</propertyname>
>   <operator>FROM_TO</operator>
>   <queryvalues>
>    <queryvalue>
>     <valueinterval>
>      <valuefrom>100000</valuefrom>
>      <valueto>120000</valueto>
>     </valueinterval>
>    </queryvalue>
>   </queryvalues>
>     </item>
>     <item>
>   <propertyname>active</propertyname>
>   <operator>EQUAL</operator>
>   <queryvalues>
>    <queryvalue>
>     <value>
>      <value>true</value>
>     </value>
>    </queryvalue>
>   </queryvalues>
>     </item>
>  <item>
>   <propertyname>characteristic</propertyname>
>   <operator>EQUAL</operator>
>   <queryvalues>
>    <queryvalue>
>     <value>
>      <!-- This is the link to the agents characteristic  -->
>
<value>com.flowfact.server/api/rest/v1.0/customers/FlowFact/characterist
ics/ED42A4EF-B82C-440A-8FB0-8F12F87BCB45</value>
>     </value>
>    </queryvalue>
>    <queryvalue>
>     <value>
>      <!-- This is the characteristic that labels an inquiry as one of
the agents department -->
>
<value>com.flowfact.server/api/rest/v1.0/customers/FlowFact/characterist
ics/CDD29B93-1994-4A71-97B5-49FBCF7DCCBC</value>
>     </value>
>    </queryvalue>
>   </queryvalues>
>     </item>
> </query>
```

When this request is sent, the response will contain a list of matching inquiries for the given parameters. When querying for inquiries you have the choice between two representations: Either you are only interested in a short summary of matching query results. In that case you may query for a simple application/xml representation. Or you are interested in a fully blown representation of the returned values. In that case you have to use the Accept: application/full+xml HTTP header information.

The response contains, depending on the FlowFact System, a list of links and some meta information about the found inquiries.

XML-Schema (XSD)

Message descriptions

Ypu can use the following XML-Schema files to generate messages that are understood by the FlowFact API.

	Version 1.0.x
Access Control List:	rest-acls-1.0.xsd
Base:	rest-basetypes-1.0.xsd
Activities:	rest-baseactivity-1.0.xsd
Characteristics:	rest-characteristics-1.0.xsd
Contacts:	rest-contacts-1.0.xsd
Duplicates:	rest-duplicates-1.0.xsd
Estates:	rest-baseestate-1.0.xsd
Inquiries:	rest-inquiries-1.0.xsd
Messages:	rest-messages-1.0.xsd
Projects:	rest-projects-1.0.xsd
Users:	rest-users-1.0.xsd
Queries:	rest-query-1.0.xsd

For further information to the customizable properties of the entities please check Special Resources->Blueprint and the documentation of the entity.