Communication Channel Subdomain

Table of Contents

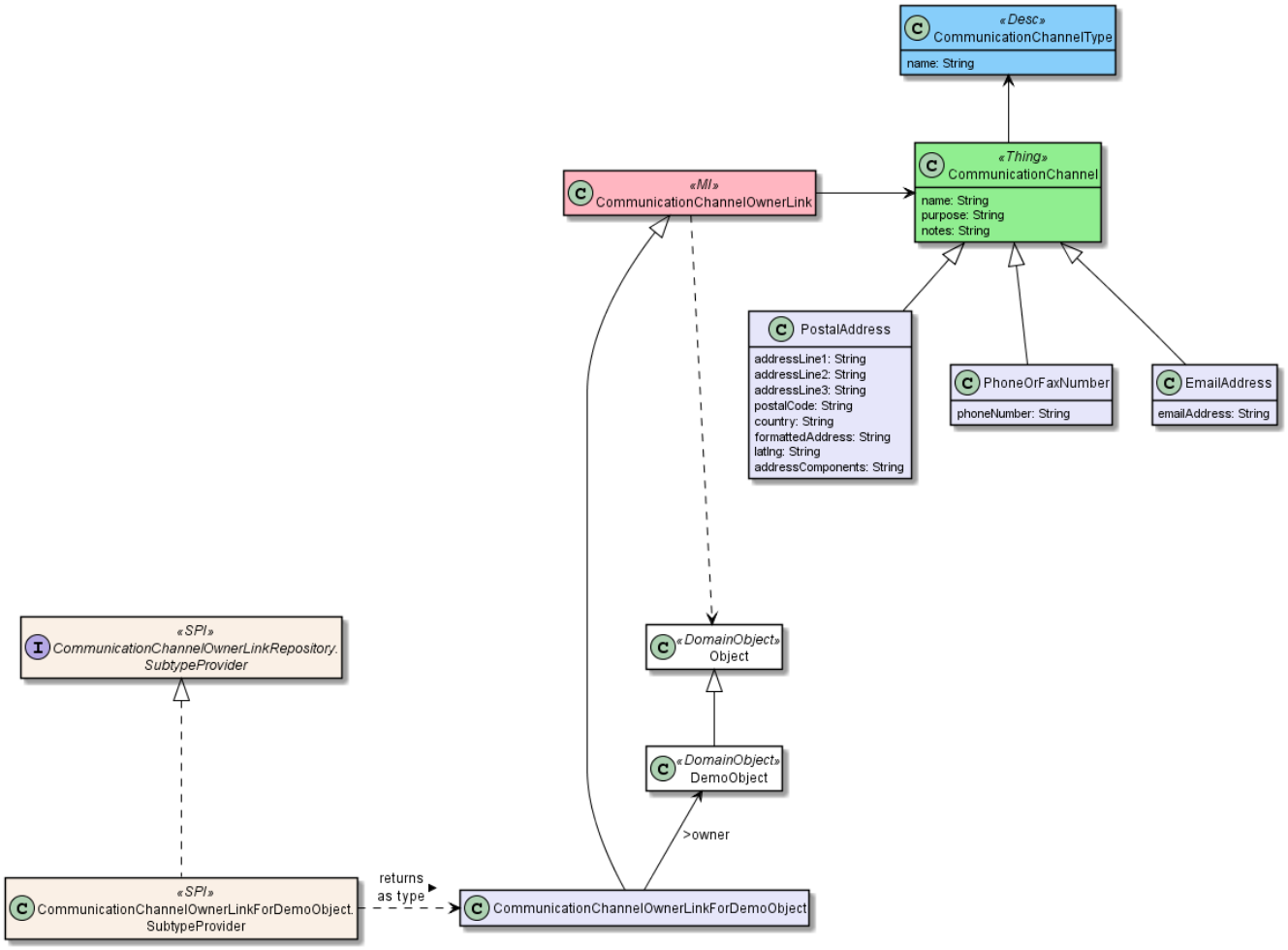
This module (`isis-module-commchannel`) provides the ability to attach communication channel objects (postal address, email or phone/fax number) to arbitrary domain entities. There are *no* requirements for those objects to implement any interfaces. A subclass of `CommunicationChannelOwnerLink` abstract class is required (about 50 lines of boilerplate), acting as the "glue" between the communication channel and its associated "owner" domain object.

For postal address, the geocoding data can be looked up by invoking the Google Geocoding API. This allows free-form text entry of an address to be converted into its geocoded equivalent, including latitude and longitude. This in turn allows each postal address object to implement the `Location` interface (from Gmap3 wicket component) meaning that they can be rendered in a map view.

Each communication channel object can optionally be categorized with a predefined "purpose", where the available list of "purposes" is defined by the optional `CommunicationChannelPurposeRepository` SPI service. The list of purposes can vary by both the communication channel type (postal address/email/phone) and also by the "owner" domain object; for example address objects the list could be "billing address" and "shipping address", whereas for emails the list of purposes might be "work email" or "home email".

Domain Model

The following class diagram highlights the main concepts:

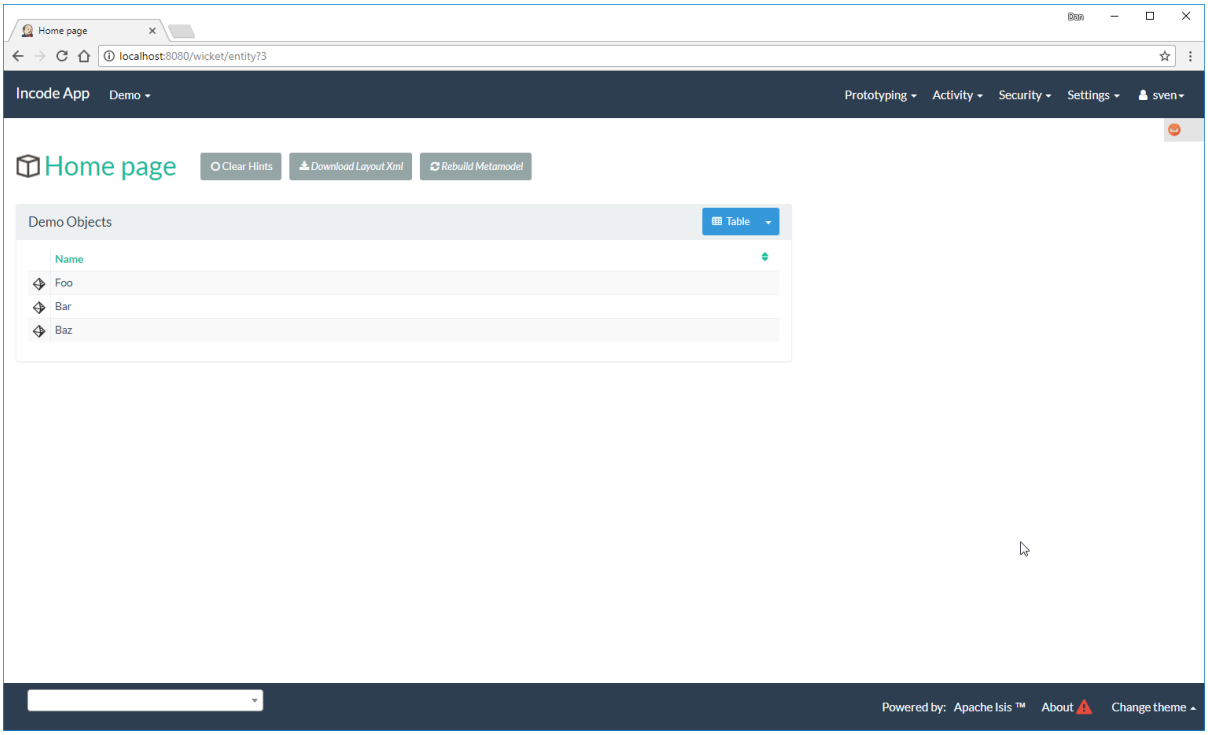


(The colours used in the diagram are - approximately - from Object Modeling in Color).

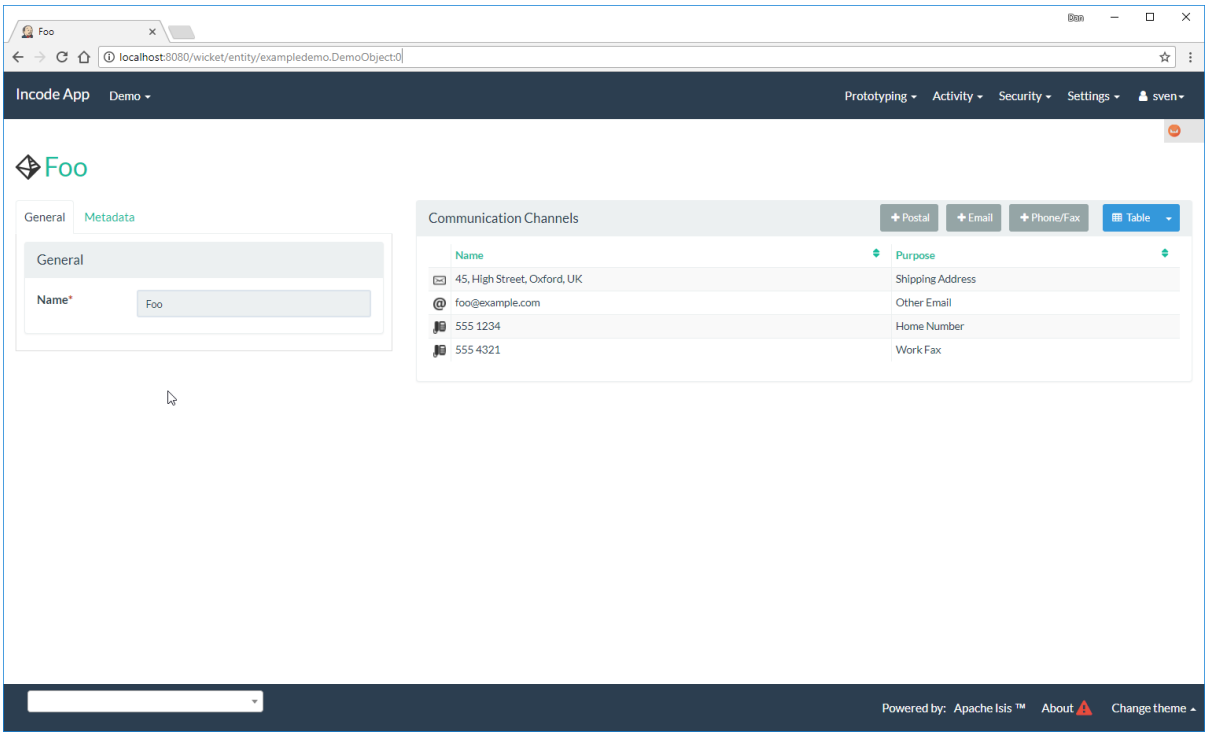
Screenshots

The module's functionality can be explored by running the quickstart with example usage using the `org.incode.domainapp.example.app.modules.ExampleDomDomCommChannelAppManifest`.

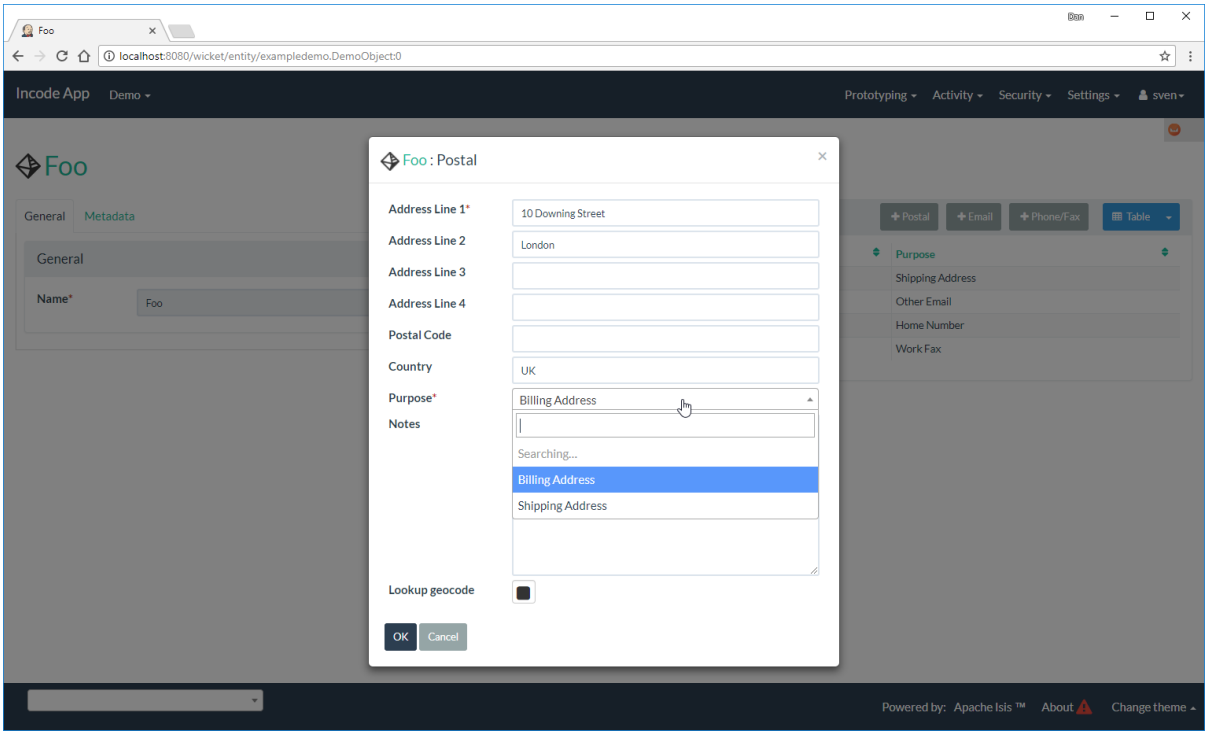
A home page is displayed when the app is run:



The fixture data sets up a number of demo objects, along with some sample `CommunicationChannel`s associated with each. These are shown in the contributed `communicationChannels` collection:

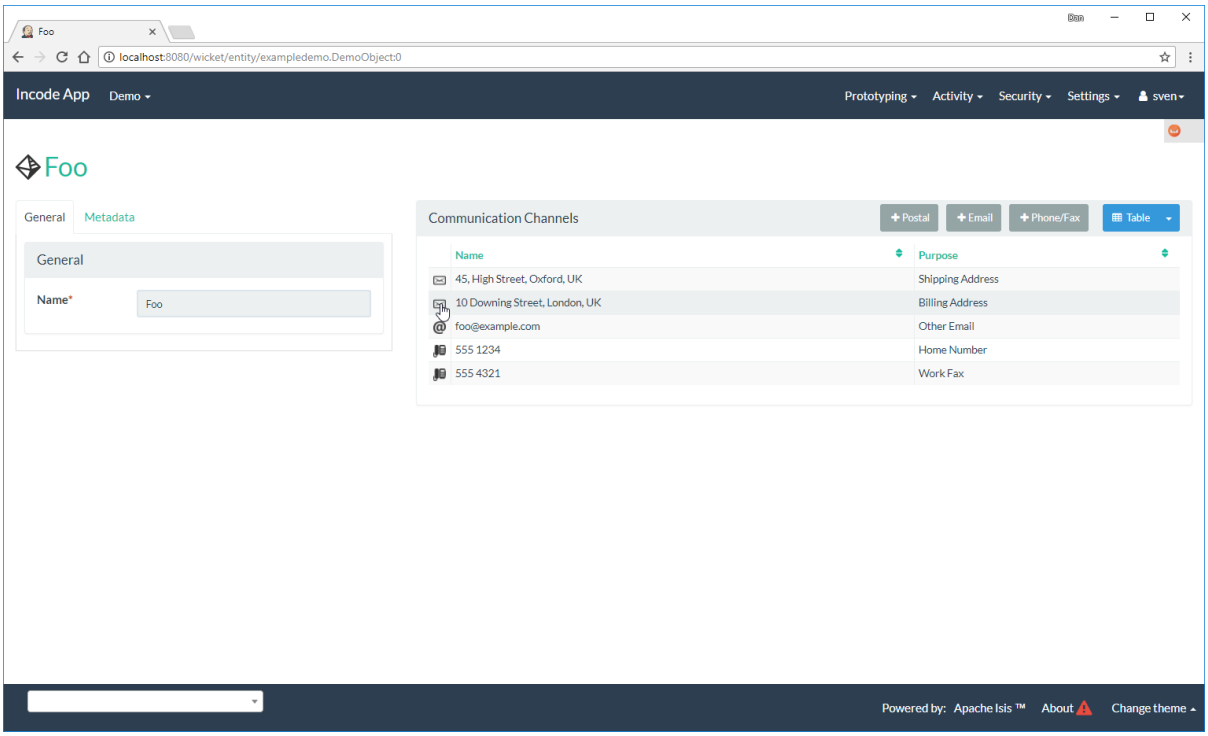


Also contributed are actions to create new communication channels of each of the three main types. For example, we can create a new `PostalAddress`:

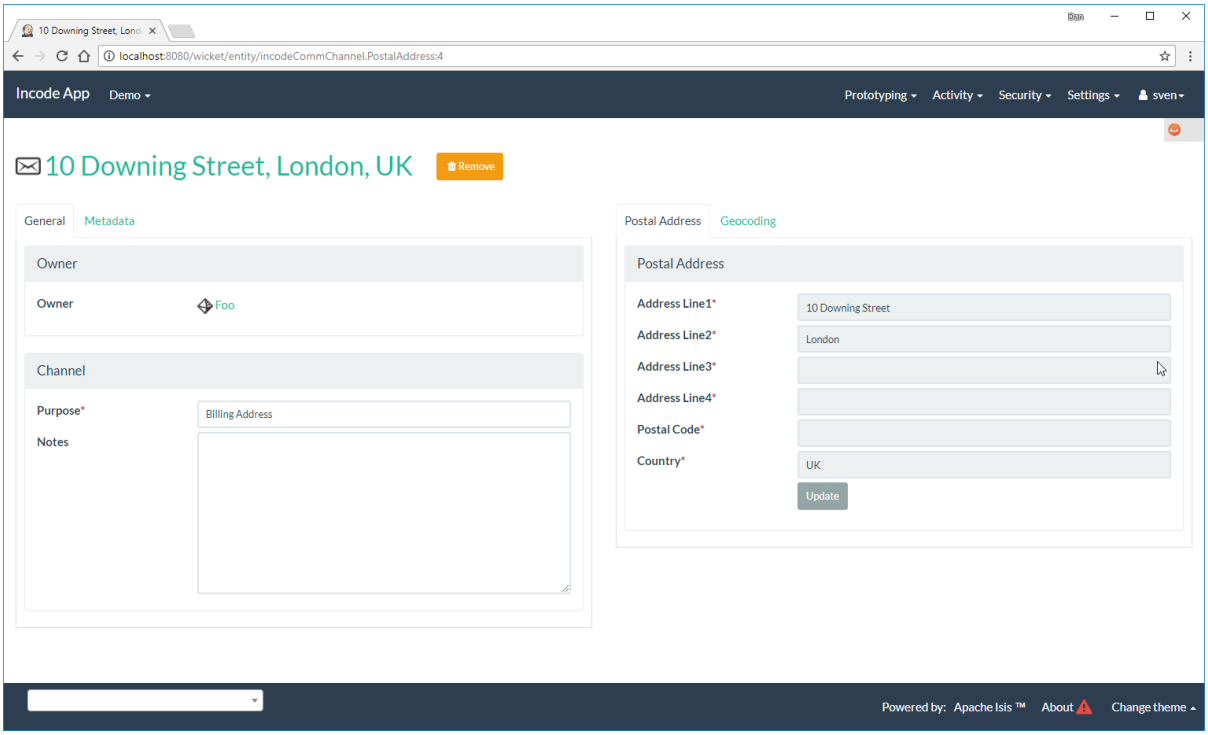


Because postal addresses can vary so much around the world, the module defines only "address lines" (rather than house street number, street address, district and so on). Apart from the first address line, these are all optional.

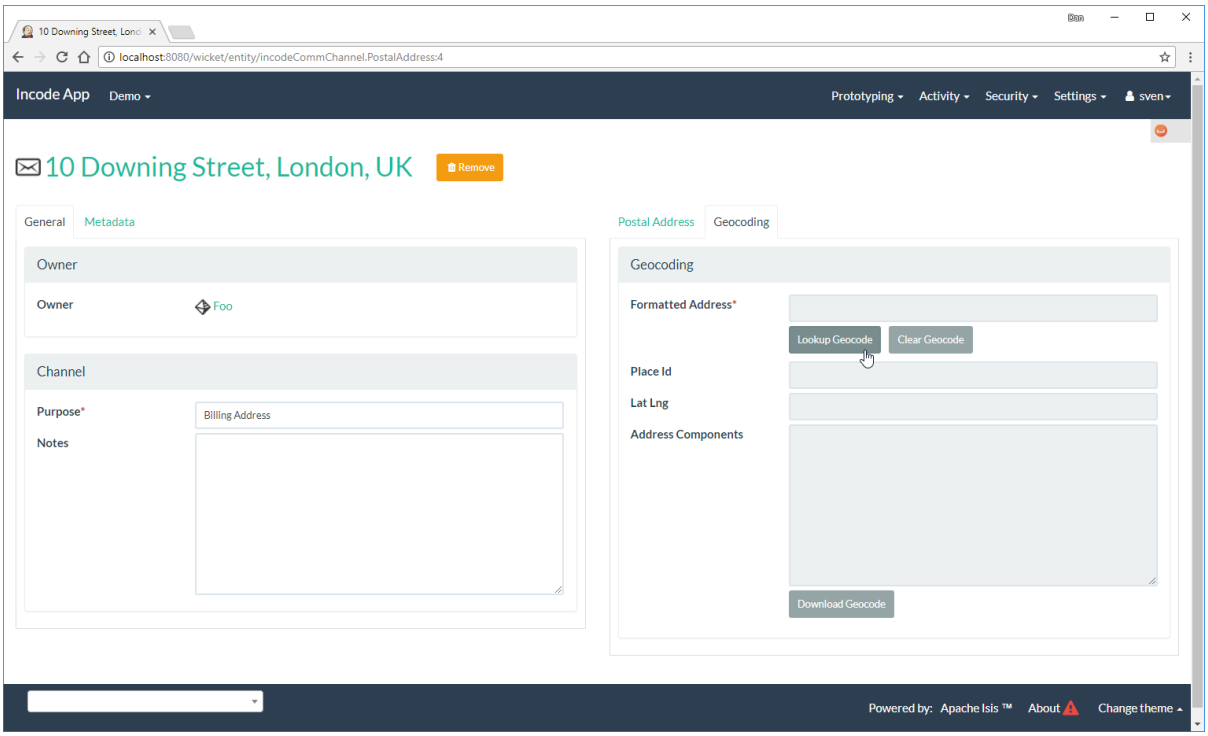
Completing the action updates the list of communication channels for the communication channel owner:



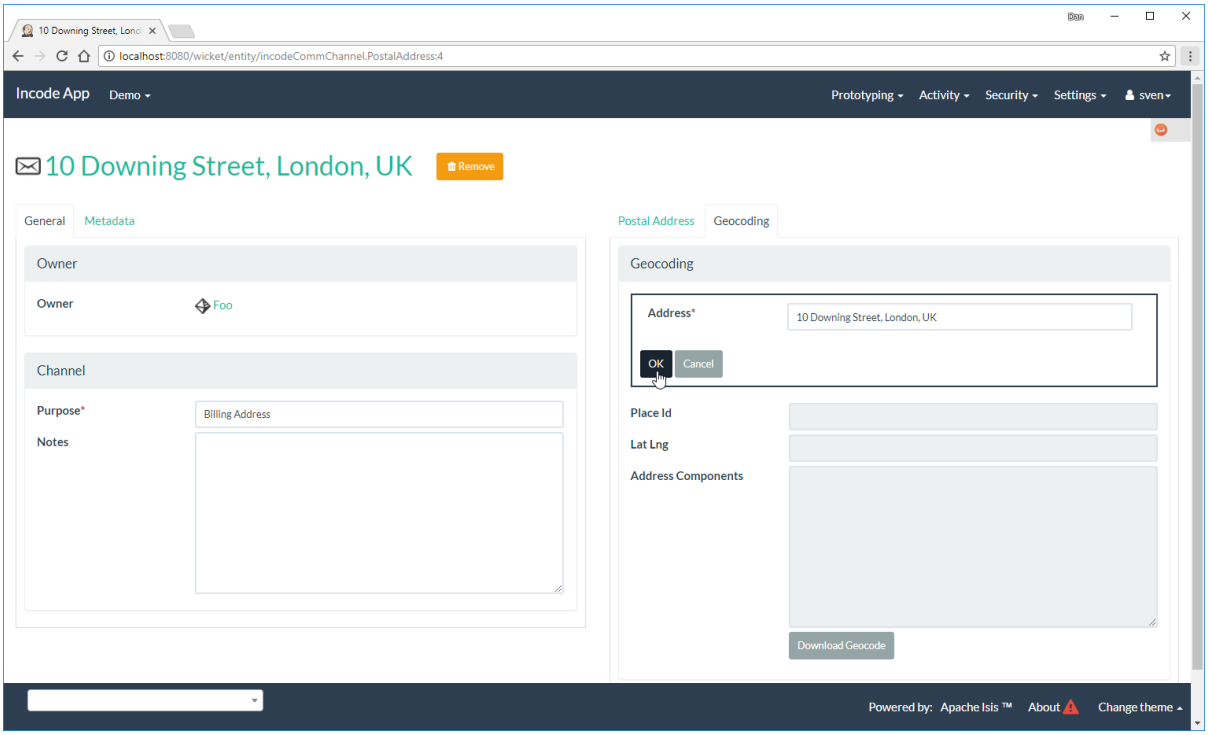
Each communication channel can be viewed and updated. The `PostalAddress` is the most complex:



Specifically, the postal address allows geocoding data to be looked up from the Google Geocoding API:

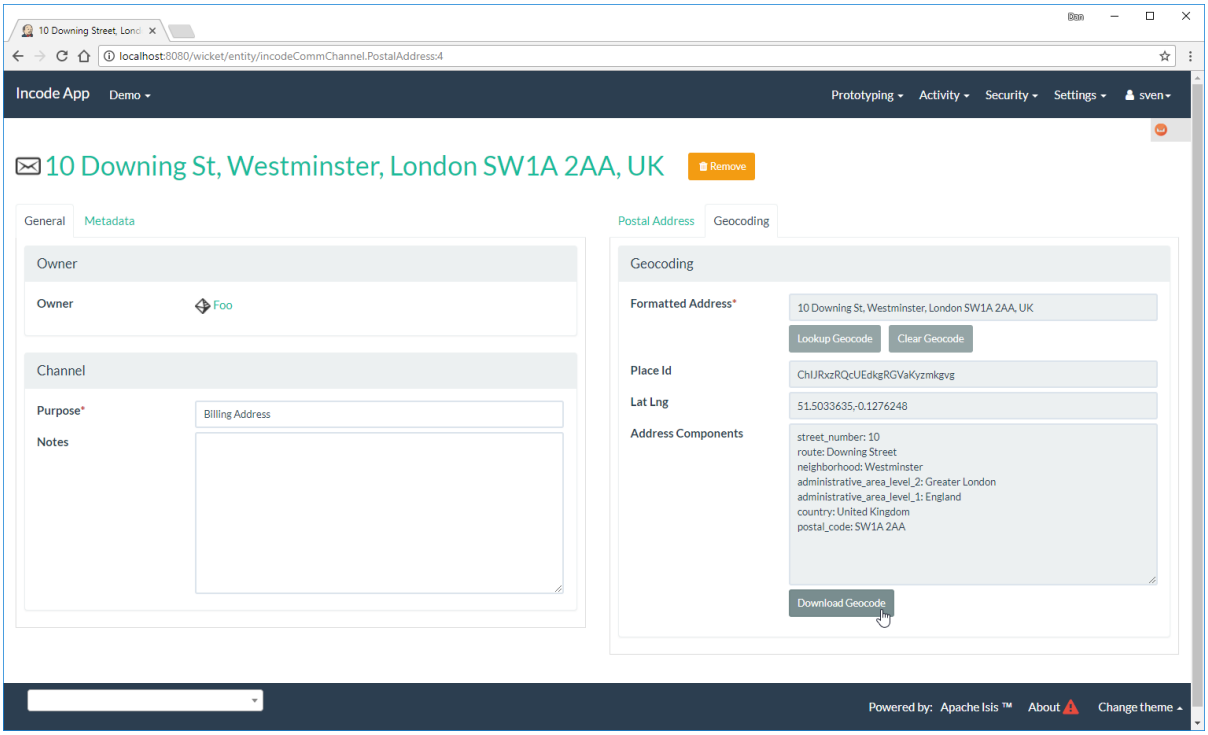


The address lines information entered previously is used for the search, but this can be adjusted as necessary by the end-user:

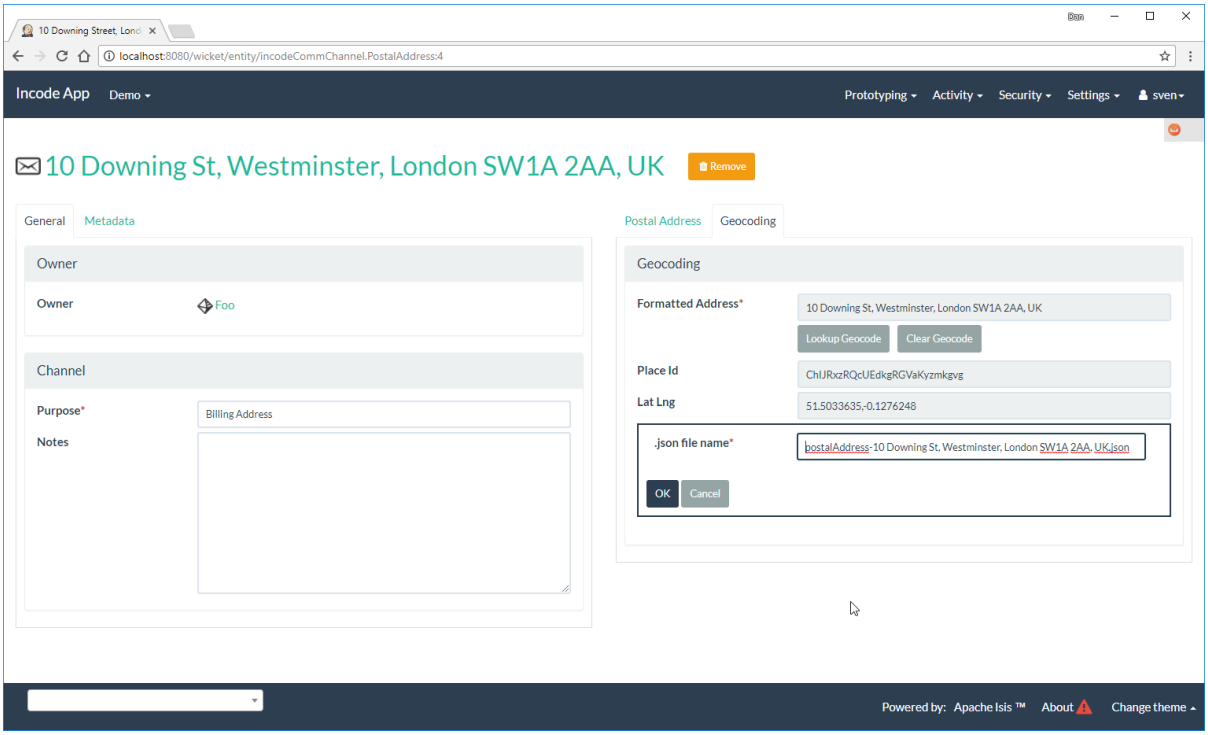


The geocoding data consists of the latitude and longitude, but also of a unique place Id. It also contains various other more detailed information, such as the various components that make up the address.

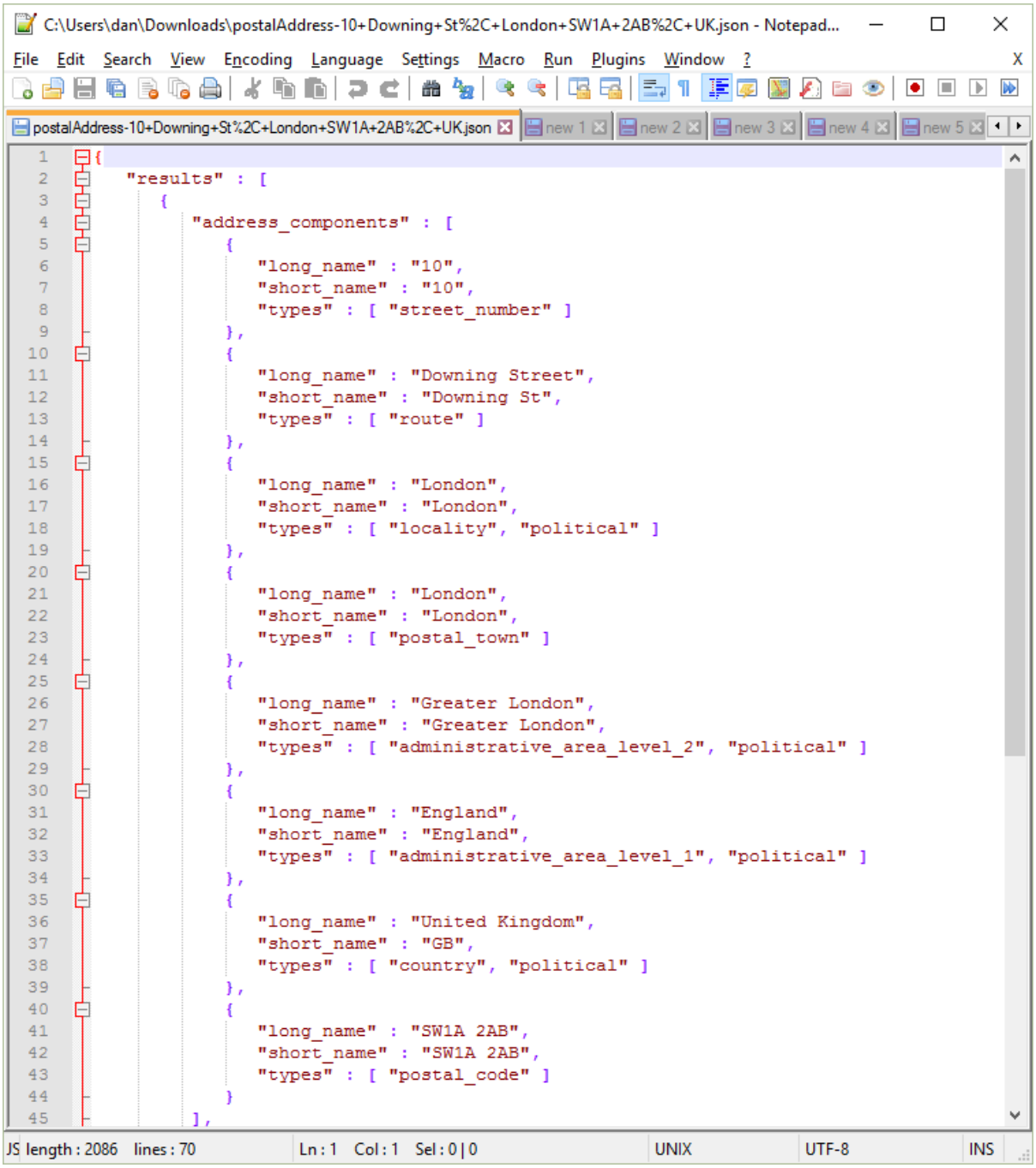
In addition, the raw JSON from the geocoding API request can be downloaded:



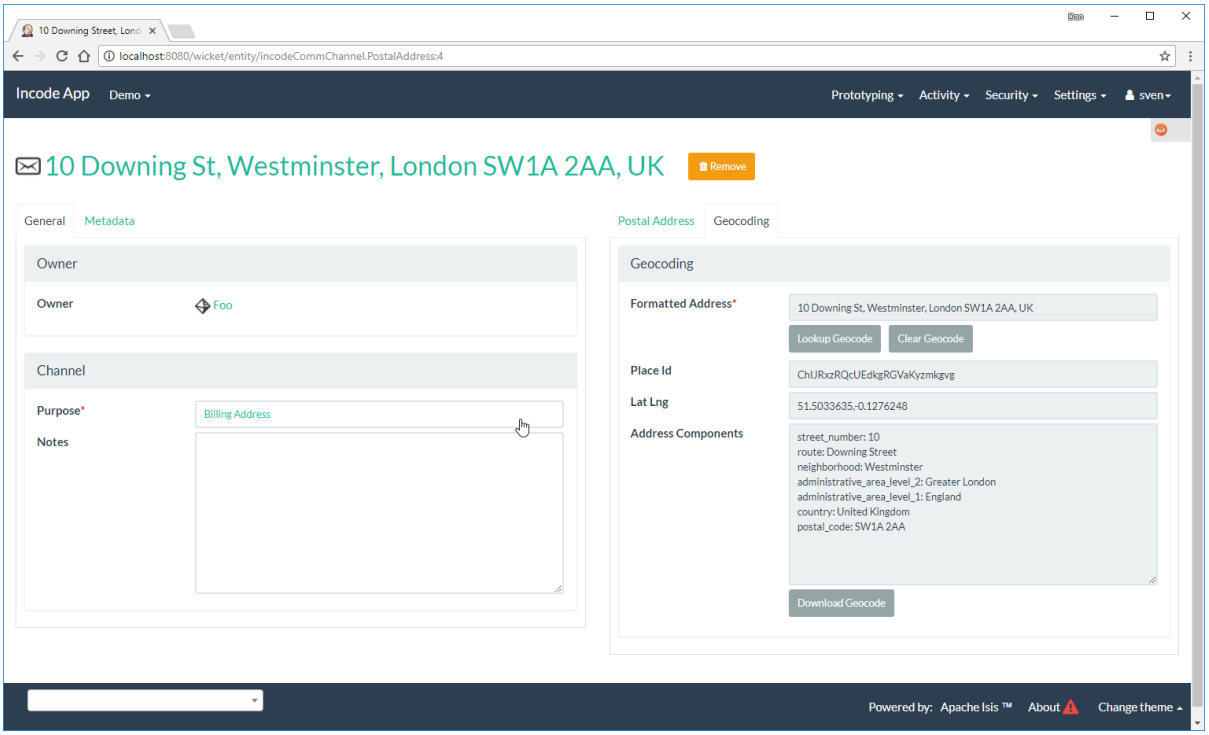
specifying a filename:


10 Downing Street, Lond
localhost:8080/wicket/entity/incodeCommChannel.PostalAddress:4
Incode App
Demo
Prototyping
Activity
Security
Settings
sven
10 Downing St, Westminster, London SW1A 2AA, UK
Remove
General
Metadata
Postal Address
Geocoding
Owner
Owner
Foo
Channel
Purpose*
Billing Address
Notes
Geocoding
Formatted Address*
10 Downing St, Westminster, London SW1A 2AA, UK
Lookup Geocode
Clear Geocode
Place Id
ChIJRxzRQcUEdkgRGVaKyzmkgvg
Lat Lng
51.5033635,-0.1276248
.json file name*
postalAddress-10 Downing St, Westminster, London SW1A 2AA, UK.json
OK
Cancel
Powered by: Apache Isis ™
About
Change theme

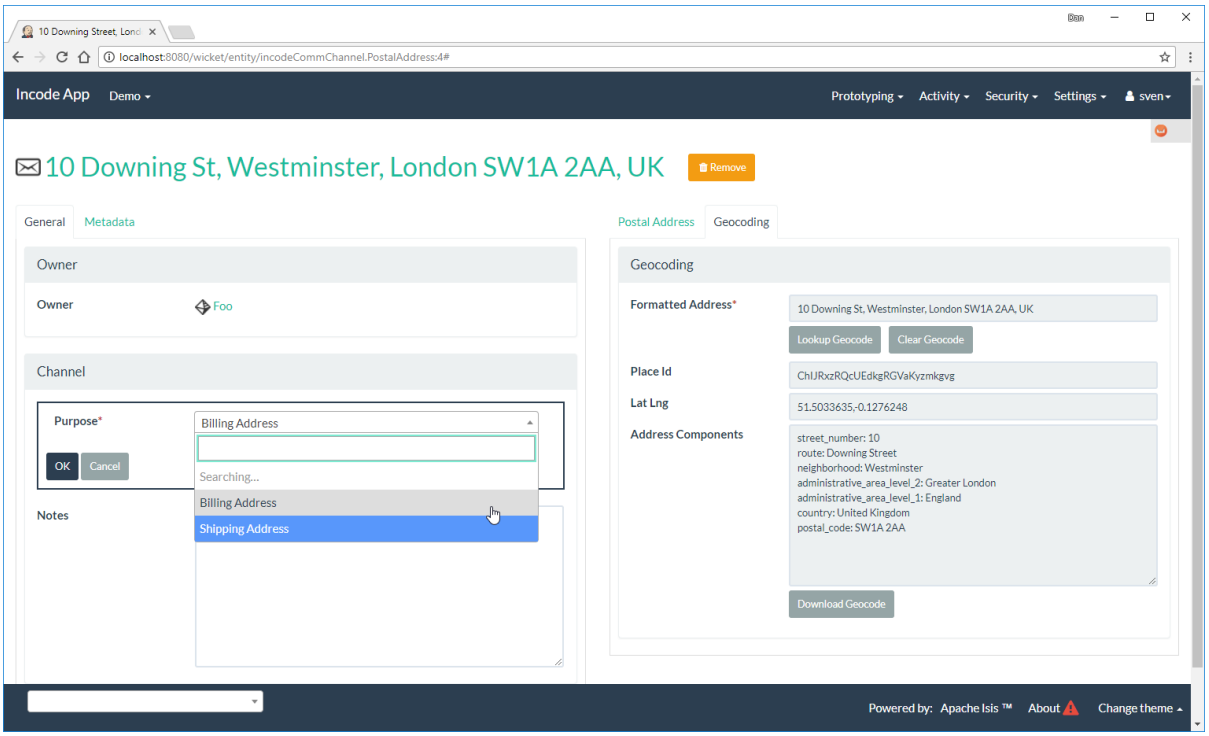
and viewed:



All communication channel have a "purpose", which can be updated:

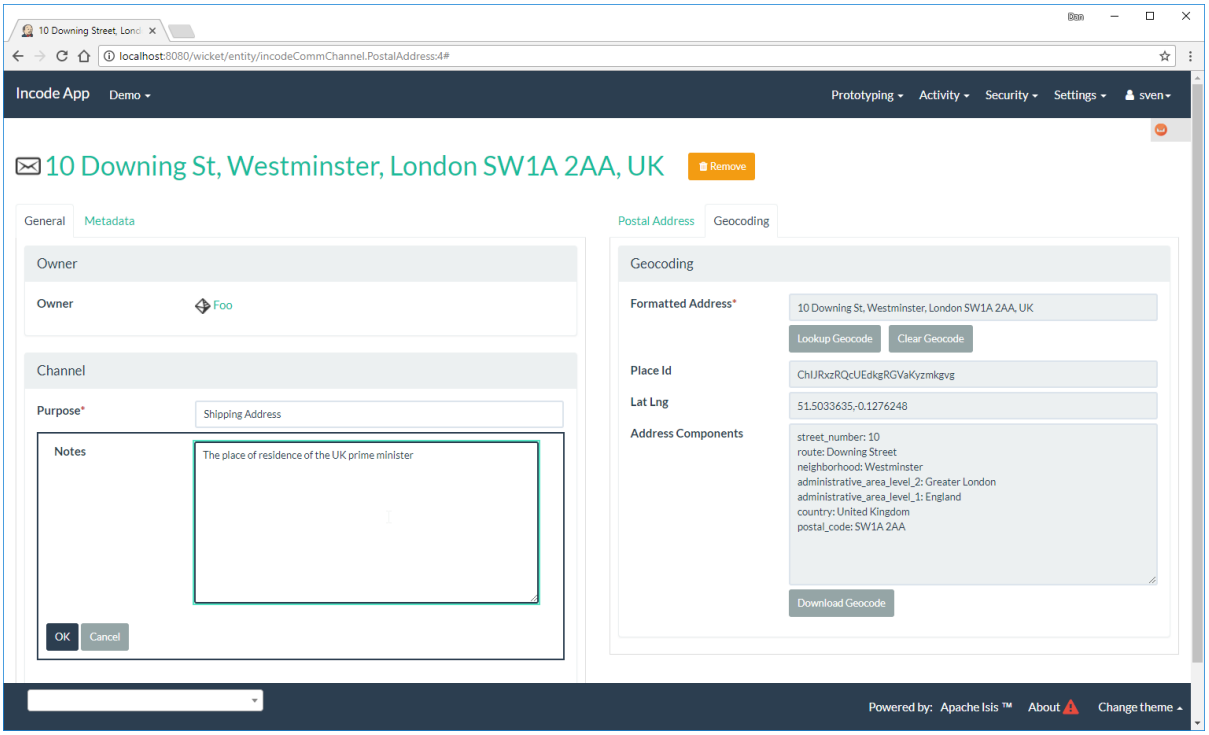


The list of available purposes varies by channel type, for example "Billing Address" or "Shipping Address" for `PostalAddress`es, "Home Email" or "Work Email" for `EmailAddress`es, etc:

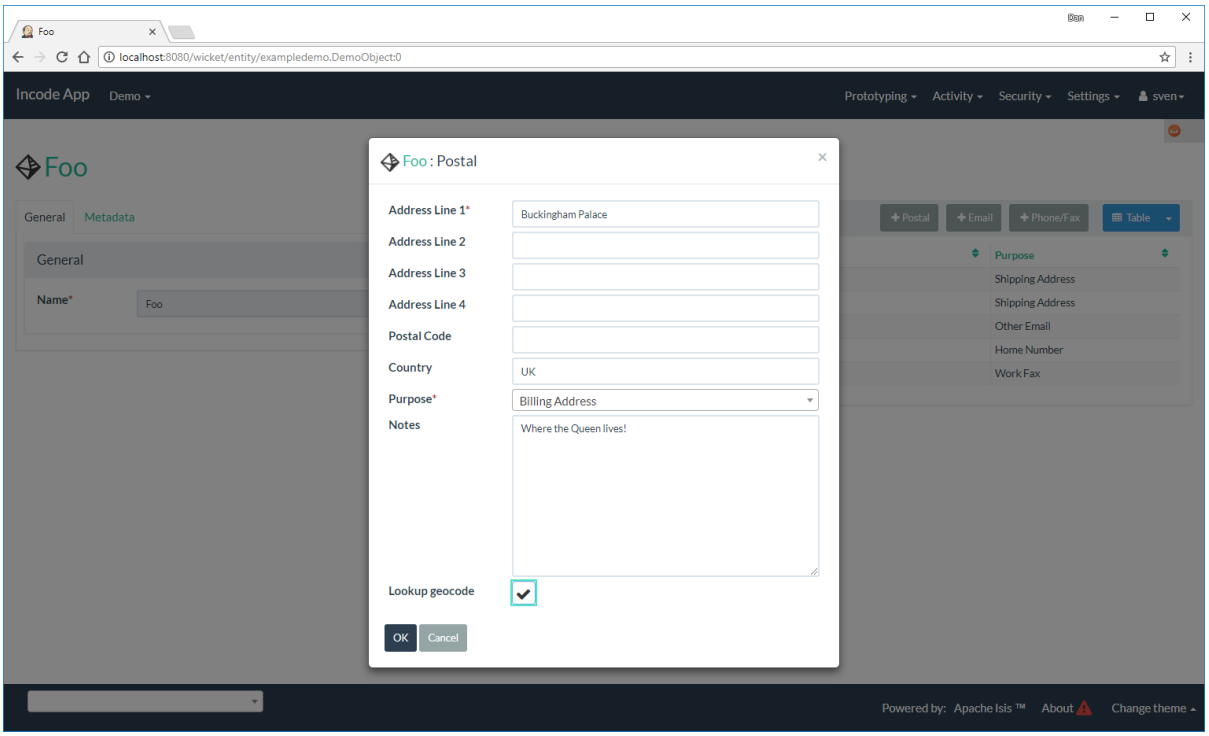


The list itself is not hardcoded into the module, however; instead it is defined by the `CommunicationChannelPurposeRepository` optional SPI. If there is no implementation of this SPI service then a default "purpose" is used

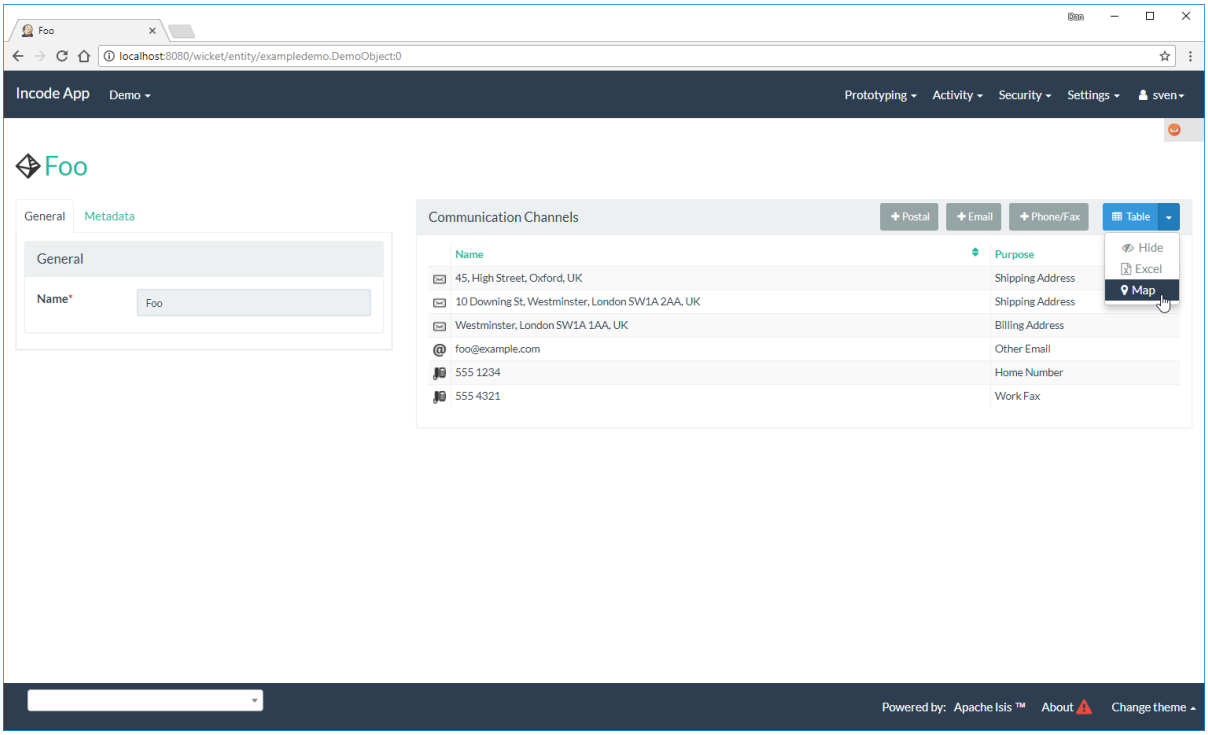
Every communication channel also allows adhoc notes to be added:



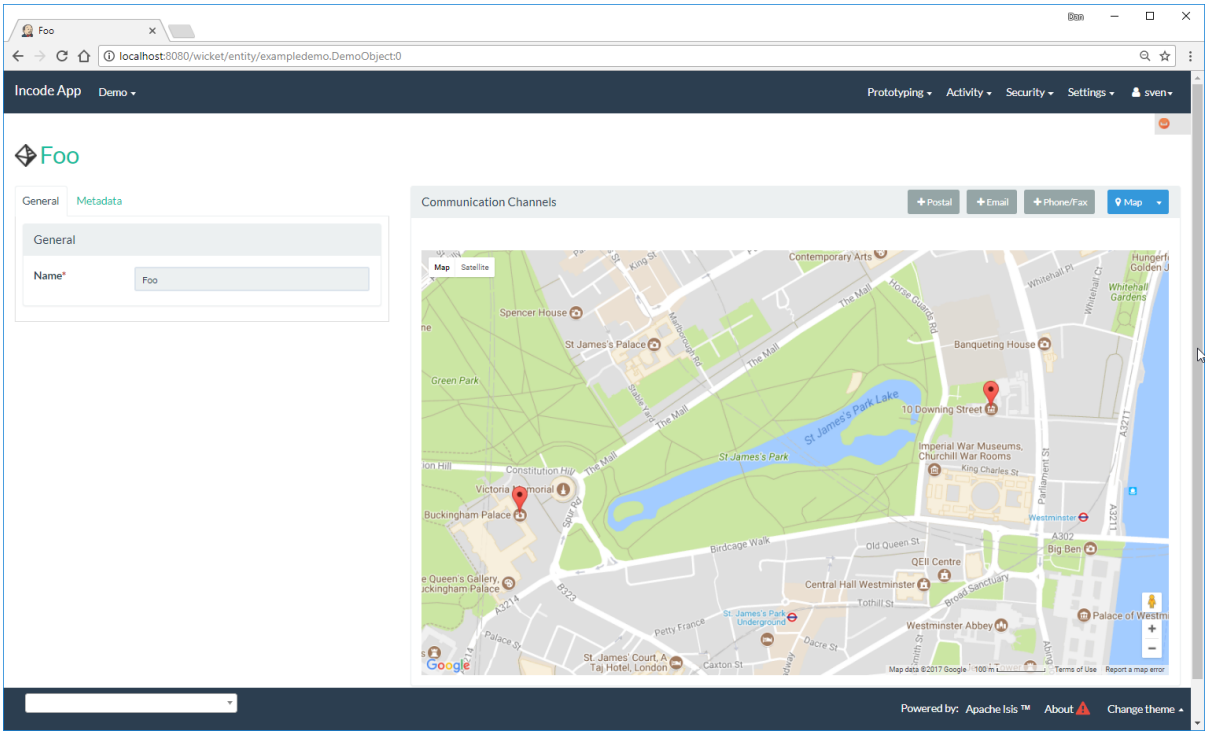
When creating a new `PostalAddress`, the geocode information can be looked up at the same time using the final checkbox parameter for the action:



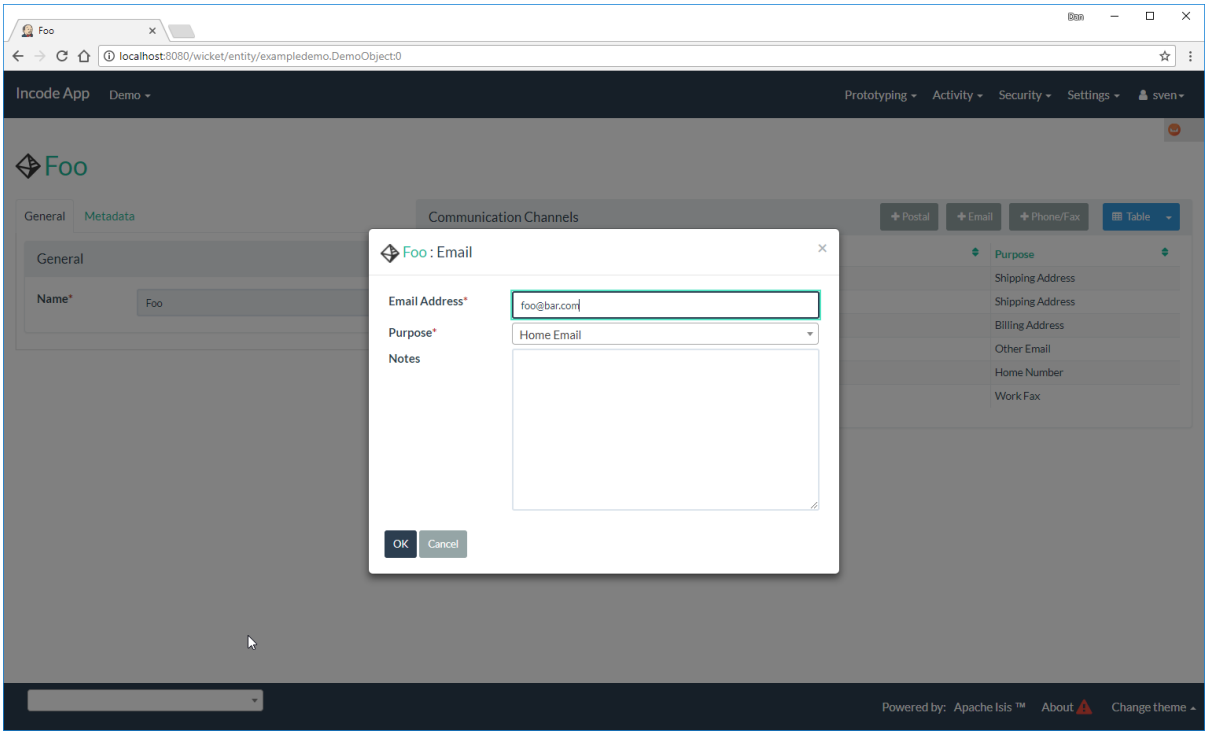
The `PostalAddress` entity implements the gmap3 component's `Locatable` interface, meaning that it can be rendered on a map. Assuming that the extension has been configured on the classpath:



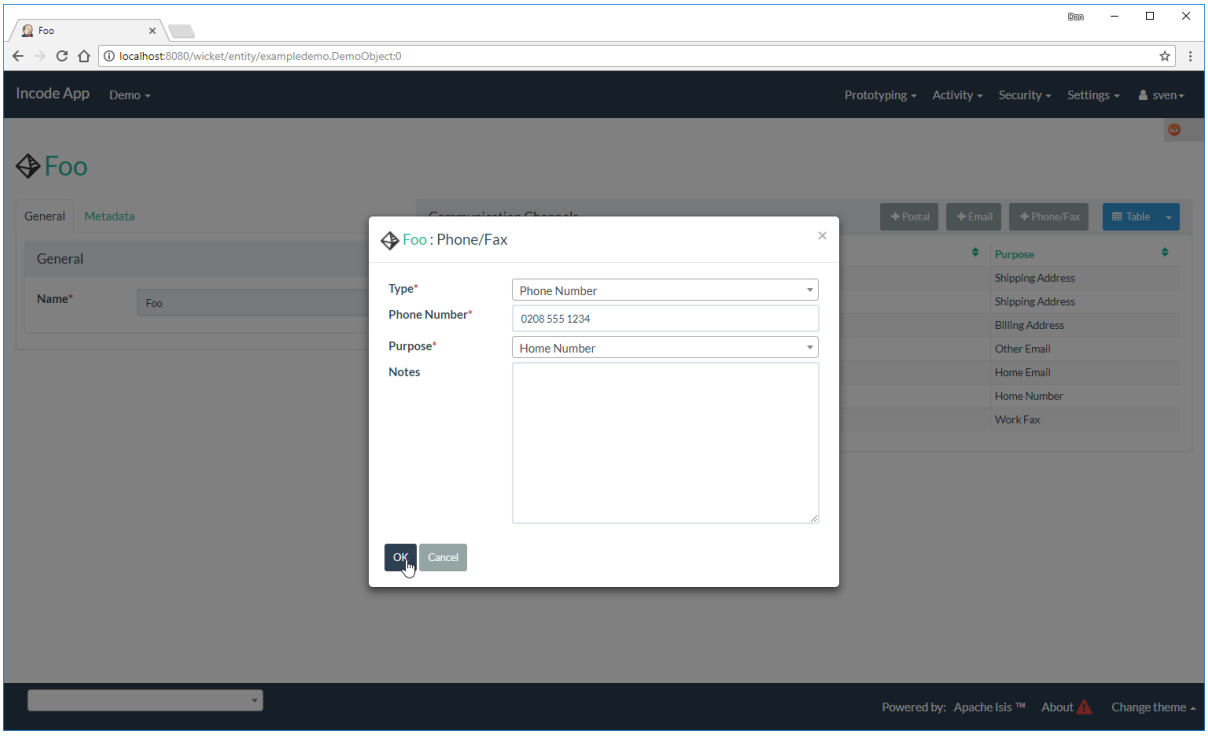
Then the map is rendered:



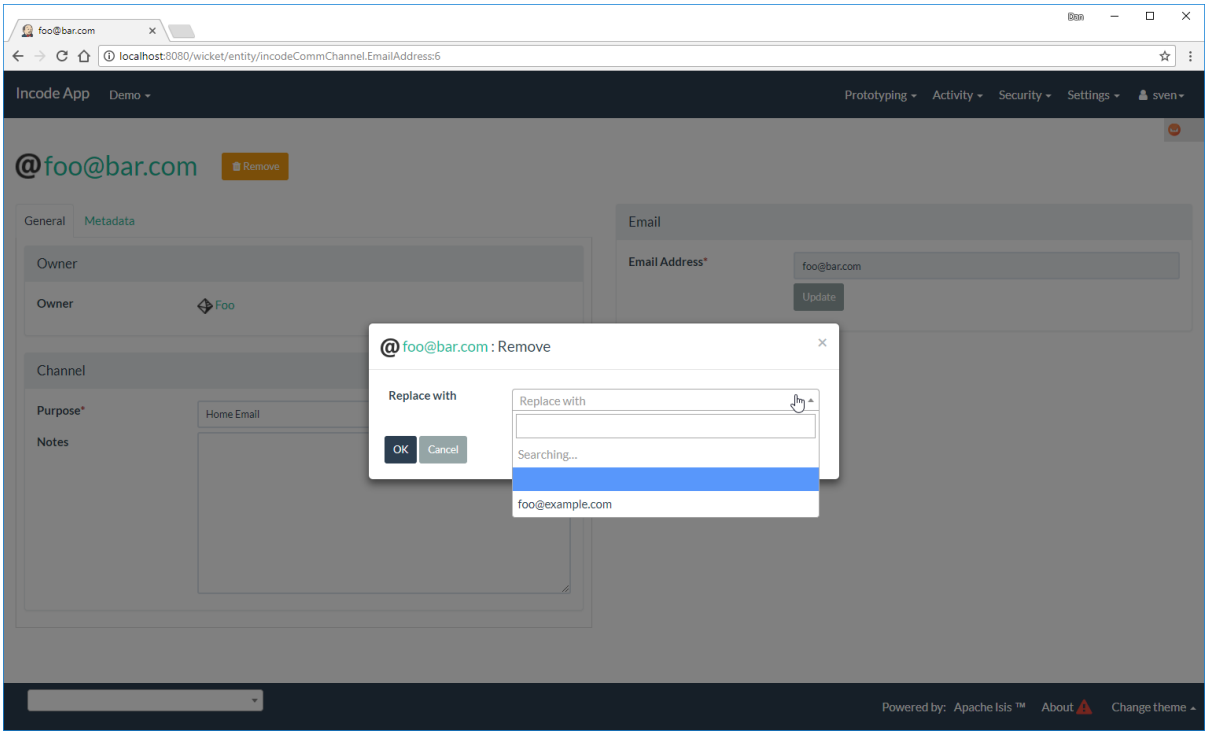
The module also allows new `EmailAddress`es to be created:



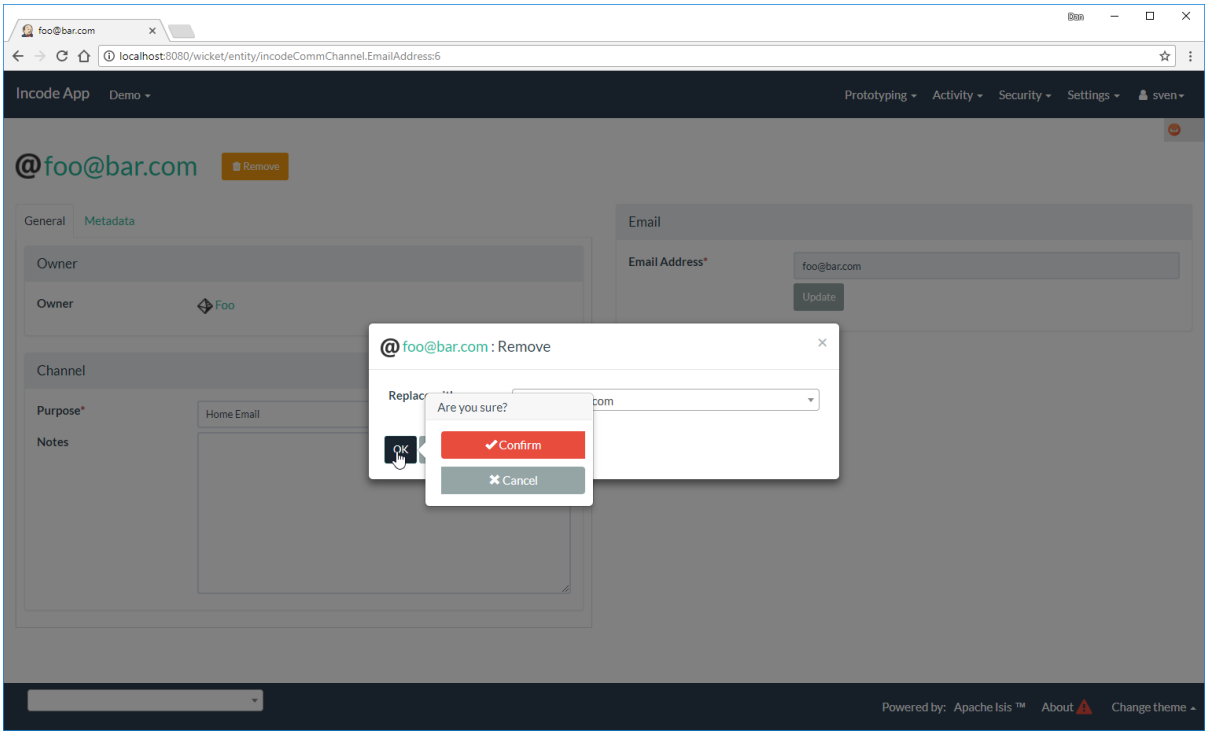
And similarly allows new `PhoneOrFaxNumber`s to be created:



What can be created and modified can also be deleted. Each of the `CommunicationChannel` objects provide an action to remove them:



As the screenshot shows, the end-user has the option of specifying some other communication channel (of the same type) as a "replacement". Because each action has a corresponding domain event, this allows for cascade updates scenarios (where other objects that depend upon the channel being deleted can instead refer to its replacement).



Alternatively, subscribers can opt to veto the removal/deletion of a communication channel. This is done using the standard technique described in the Apache Isis user guide.

How to configure/use

Classpath

Update your classpath by adding this dependency in your dom project's `pom.xml`

```
<dependency>
    <groupId>org.incode.module.commchannel</groupId>
    <artifactId>incode-module-commchannel-dom</artifactId>
    <version>1.15.0</version>
</dependency>
```

Check for later releases by searching Maven Central Repo.

For instructions on how to use the latest `-SNAPSHOT`, see the contributors guide.

Bootstrapping

In the `AppManifest`, update its `getModules()` method:

```
@Override
public List<Class<?>> getModules() {
    return Arrays.asList(
            ...
            org.incode.module.commchannel.dom.CommChannelModule.class,
    );
}
```

For each domain object...

In order to be able to add/remove communication channels for a domain object, you need to:

- implement a subclass of `CommunicationChannelOwnerLink` for the domain object's type.

  This link acts as a type-safe tuple linking the domain object to the `CommunicationChannel`.

- implement the `CommunicationChannelOwnerLinkRepository.SubtypeProvider` SPI interface:

  ```
  public interface SubtypeProvider {
      Class<? extends CommunicationChannelOwnerLink> subtypeFor(
          Class<?> domainObject,
          CommunicationChannelType communicationChannelType);
  }
  ```

  This tells the module which subclass of `CommunicationChannelOwnerLink` to use to attach to the

domain object. The `SubtypeProviderAbstract` adapter can be used to remove some boilerplate.

- subclass `T_addEmailAddress`, `T_addPostalAddress`, `T_addPhoneOrFaxNumber` and `T_communicationChannels` (abstract) mixin classes for the domain object.

  These contribute the "communication channels" collection and actions to add communication channels of the various types.

Typically the SPI implementations and the mixin classes are nested static classes of the `CommunicationChannelOwnerLink` subtype.

For example, in the demo app the `CommChannelDemoObject` domain object can "own" communication channels by virtue of the `CommunicationChannelOwnerLinkForDemoObject` subclass:

```
@javax.jdo.annotations.PersistenceCapable(identityType= IdentityType.DATASTORE,
schema="incodeCommChannelDemo")
@javax.jdo.annotations.Inheritance(strategy = InheritanceStrategy.NEW_TABLE)
@DomainObject
public class CommunicationChannelOwnerLinkForDemoObject extends
CommunicationChannelOwnerLink { ①

    private CommChannelDemoObject demoObject;
    @Column( allowsNull = "false", name = "demoObjectId")
    public CommChannelDemoObject getDemoObject() {
②
        return demoObject;
    }
    public void setDemoObject(final CommChannelDemoObject demoObject) {
        this.demoObject = demoObject;
    }

    public Object getOwner() {
③
        return getDemoObject();
    }
    protected void setOwner(final Object object) {
        setDemoObject((CommChannelDemoObject) object);
    }

    @DomainService(nature = NatureOfService.DOMAIN)
    public static class SubtypeProvider
            extends CommunicationChannelOwnerLinkRepository.SubtypeProviderAbstract {
④
        public SubtypeProvider() {
            super(CommChannelDemoObject.class,
CommunicationChannelOwnerLinkForDemoObject.class);
        }
    }

    @Mixin
```

```
    public static class _communicationChannels
⑤
            extends T_communicationChannels<CommChannelDemoObject> {
        public _communicationChannels(final CommChannelDemoObject owner) {
            super(owner);
        }
    }
    @Mixin
    public static class _addEmailAddress extends T_addEmailAddress
<CommChannelDemoObject> {
        public _addEmailAddress(final CommChannelDemoObject owner) {
            super(owner);
        }
    }
    @Mixin
    public static class _addPhoneOrFaxNumber extends T_addPhoneOrFaxNumber
<CommChannelDemoObject> {
        public _addPhoneOrFaxNumber(final CommChannelDemoObject owner) {
            super(owner);
        }
    }
    @Mixin
    public static class _addPostalAddress extends T_addPostalAddress
<CommChannelDemoObject> {
        public _addPostalAddress(final CommChannelDemoObject owner) {
            super(owner);
        }
    }
}
```

① extend from `CommunicationChannelOwnerLink`

② the type-safe reference property to the "owning" domain object (in this case `DemoObject`). In the RDBMS this will correspond to a regular foreign key with referential integrity constraints correctly applied.

③ implement the hook `setOwner(…)` method to allow the type-safe reference property to the "owner" (in this case `DemoObject`) to be set. Also implemented `getOwner()` similarly.

④ implementation of the `SubtypeProvider` SPI domain service, telling the module which subclass of `CommunicationChannelOwnerLink` to instantiate to attach to the owning domain object

⑤ mixins for the collections and actions contributed to the owning domain object

SPI

The `CommunicationChannelPurposeRepository` interface can optionally be implemented to specify the available "purpose" for each `CommunicationChannel` domain object.

For example, in the demo app this is implemented as:

```
@DomainService(nature = NatureOfService.DOMAIN)
public class CommunicationChannelPurposeRepositoryForDemo implements
CommunicationChannelPurposeRepository {
    @Override
    public Collection<String> purposesFor(
            final CommunicationChannelType communicationChannelType,
            final Object owner) {
        switch (communicationChannelType) {
            case EMAIL_ADDRESS:
                return Arrays.asList("Home Email", "Work Email", "Other Email");
            case POSTAL_ADDRESS:
                return Arrays.asList("Billing Address", "Shipping Address");
            case PHONE_NUMBER:
                return Arrays.asList("Home Number", "Work Number", "Mobile Number");
            case FAX_NUMBER:
                return Arrays.asList("Home Fax", "Work Fax");
        }
        return null;
    }
}
```

If no implementation of this interface can be found, then the module provides a single "default" purpose for all communication channels.

UI Concerns

Suppressing/adding UI elements

Every property, collection and action has a corresponding domain event. Thus, a subscriber can be used to hide or disable UI representation of any domain object's members.

For example, the "notes" property could be suppressed using the following service:

```
@DomainService(nature = NatureOfService.DOMAIN)
public class CommChannelDemoSuppressNotesSubscriber extends AbstractSubscriber {
    @Subscribe
    public void on(CommunicationChannel.NotesDomainEvent ev) {
        switch (ev.getEventPhase()) {
        case HIDE:
            // uncomment as an example of how to influence the UI
            // (the 'note' property should disappear)
            // ev.hide();
        }
    }
}
```

Conversely, new UI elements can be added using contributions and mixins.

Link class

The `CommunicationChannelOwnerLink` object is not intended to be rendered directly in the UI. Rather, the `T_communicationChannels` mixin renders the referenced `CommunicationChannel`s instead.

Nevertheless (just in case there is a requirement to render the link object), the `CommunicationChannelOwnerLink` allows its title, icon and CSS class to be specified using subscribers to UI event classes specific to the link class.

Other Services

The module provides the following domain services for querying notes:

- `CommunicationChannelRepository`

  To search for `CommunicationChannel`s by owner and type.

- `CommunicationChannelOwnerLinkRepository`

  To search for `CommunicationChannelOwnerLink``s, ie the tuple that links a `CommunicationChannel` to an arbitrary `CommunicationChannelOwner`. This repository is likely to be less useful than `CommunicationChannelRepository`, but is crucial to the internal workings of the `incode-module-commchannel` module.

Related Modules/Services

The module implements the gmap3 component's `LocationDereferencingService` SPI, so that clicking on a marker on a map will render the "owning" domain object, rather than the details of the `CommunicationChannel` itself.

Known issues

None known at this time.

Dependencies

Maven can report modules dependencies using:

```
mvn dependency:list -o -pl modules/dom/commchannel/impl -D excludeTransitive=true
```

which, excluding Incode Platform and Apache Isis modules, returns these compile/runtime dependencies:

```
com.google.code.gson:gson:jar:2.3.1
org.apache.commons:commons-lang3:jar:3.1
```

From the Incode Platform it uses:

- gmap3 wicket component.

The module *also* uses icons from icons8.