RestFull Webservices aus dem Modulsystem ansteuern am Beispiel einer Online-Temperaturabfrage

Dieses Tutorial zeigt auf, wie ein Webservice auf Basis von HTTP und JSON, vom Modulsystem abgefragt werden kann. Als Beispiel wird der Online-Wetterdienst von Openweathermap.org verwendet. Der Dienst kann für eine bestimmte Anzahl von Abfragen kostenlos genutzt werden. Um das Beispiel nachzubauen, benötigt man nur eine kostenfreie Registrierung.

Vor dem Beispiel, findet Ihr noch eine Erklärung der speziellen Komponenten zur Abfrage von HTTP-Diensten.

Komponenten für RestFull Webservices

CreateHttpUrlWithParams

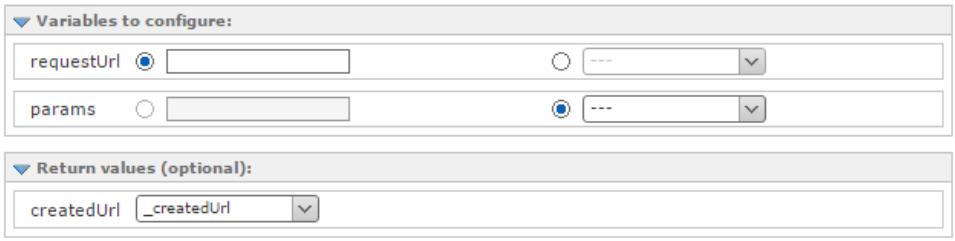
Mit dieser Komponente kann, für eine folgende HTTP-Abfrage, die benötigte URL mit Parametern zusammengesetzt werden. Die Parameter müssen vorher in einer Variable vom Typ MAP gesammelt werden.

Eigenschaften der Komponente:

- *requestUrl:* URL des Webservices
- *params:* Variable vom Typ MAP mit Parametern

Rückgabe der Komponente:

- *createdUrl:* Rückgabe der kompletten URL als Text



HttpRequest

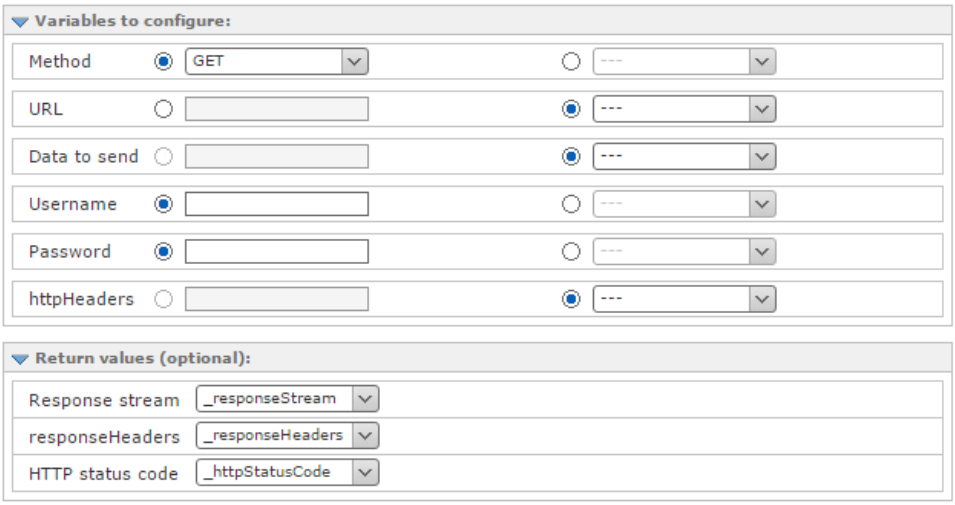
Mit dieser Komponente kann eine HTTP-Abfrage durchgeführt werden. Als HTTP-Methoden stehen GET, POST, PUT, PATCH und DELETE zur Auswahl. Die Abfrage wird synchron durchgeführt und kann individuelle Daten und Header enthalten. Zusätzlich lassen sich Zugangsdaten für eine HTTP-Authentifizierung mitsenden.

Eigenschaften der Komponente:

- *Method:* HTTP-Methode
- *URL:* Ziel-URL
- *Data to send:* Daten die übermittelt werden sollen
- *Username:* Benutzername für HTTP-Authentifizierung
- *Password:* Passwort für HTTP-Authentifizierung
- *httpHeaders:* Variable vom Typ MAP mit Header-Daten

Rückgabe der Komponente:

- *Response stream:* Inhalt der HTTP-Antwort
- responseHeaders: Header der HTTP-Antwort
- *HTTP status code:* HTTP-Statuscode als Nummer



ReadFromStream

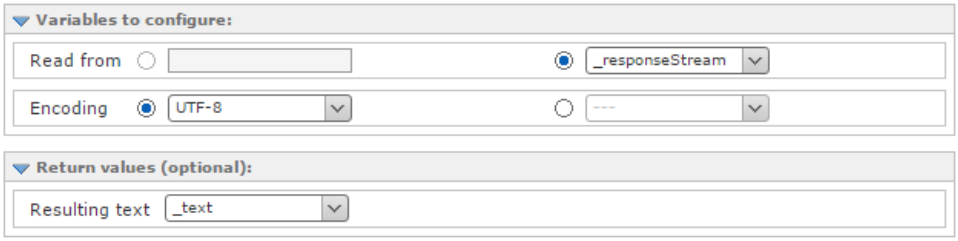
Mit dieser Komponente kann ein Daten-Stream in einen Text umgewandelt werden. Für die Umwandlung muss das Encoding (Zeichenkodierung) festgelegt werden.

Eigenschaften der Komponente:

- *Read from:* Stream der umgewandelt werden soll
- *Encoding:* Bestimmt mit welcher Encoding der Text gewandelt wird

Rückgabe der Komponente:

- *Resulting text:* Rückgabe des Textes



JsonToObject

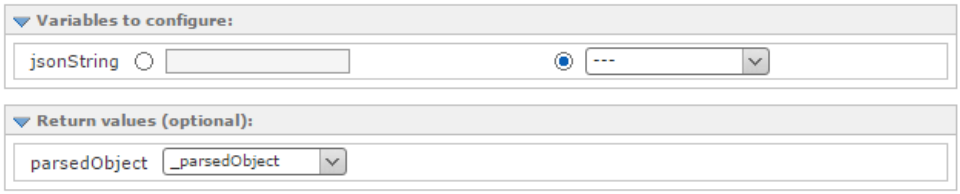
Wandelt einen Text, der JSON enthält, in ein STARFACE-Objekt um. Dabei werden JSON-Objekte zu *MAPs* und JSON-Listen zu *LISTs*.

Eigenschaften der Komponente:

- *Read from:* Stream der umgewandelt werden soll

Rückgabe der Komponente:

- *Resulting text:* Rückgabe des Textes



JSON

Die JavaScript Object Notation, kurz JSON, ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustauschs zwischen Anwendungen (Quelle: Wikipedia). Viele moderne Webservices verwenden JSON zum Austausch von Informationen. Das STARFACE Modulsystem ermöglicht es, JSON-Objekte in List und Map-Objekte zu wandeln, welche mit weiteren Komponenten verarbeitet werden können.

Aufbau eines JSON-Objektes, am Beispiel eines Büro-Objektes

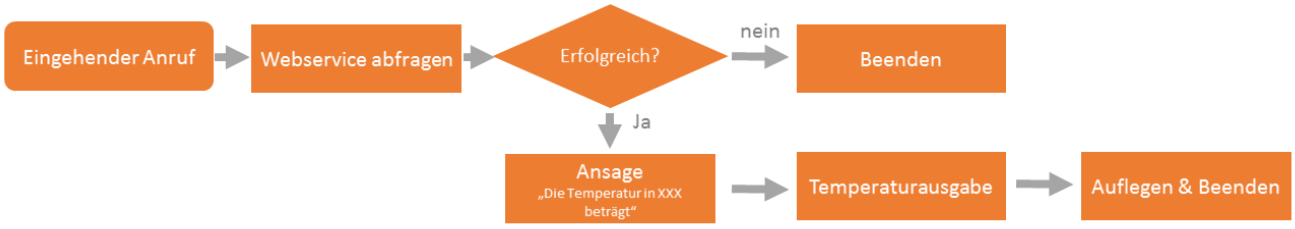
Büro-Objekt

```
{
        "Raum 1": [{
                        "Platz": "1"
                        "name": "Niklas Held",
                }, {
                        "Platz": "2"
                        "name": "Miriam Laup",
                }
        ],
        "Raum 2": [{
                        "Platz": "1"
                        "name": "Marcel
Grachner",
                }, {
                        "Platz": "2"
                        "name": "Uschi Meier",
                }, {
                        "Platz": "3"
                        "name": "Franz
Diertle",
                }
        ],
        "Küche": ["Stuhl 1", "Stuhl 2"]
}
```

Das Objekt basiert darauf, dass jeder Raum eine Eigenschaft ist, welche entweder ein weiteres Objekt (siehe Raum 1) oder eine Liste (Küche) enthält.
Die Wandlung dieses Objektes im Moduldesigner ergibt folgendes Ergebnis:

MAP	
Raum 1	**MAP** Platz: 1 name: Niklas Held ... weitere MAP für Platz 2
Raum 2	... 3 MAP für Platz 1 - 3
Küche	**Liste** Stuhl 1 Stuhl 2

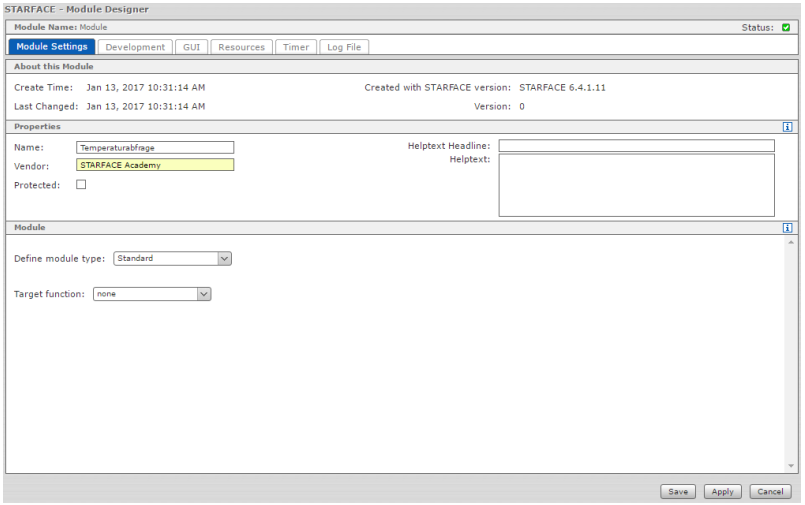
Ablaufdiagramm des Beispielmoduls



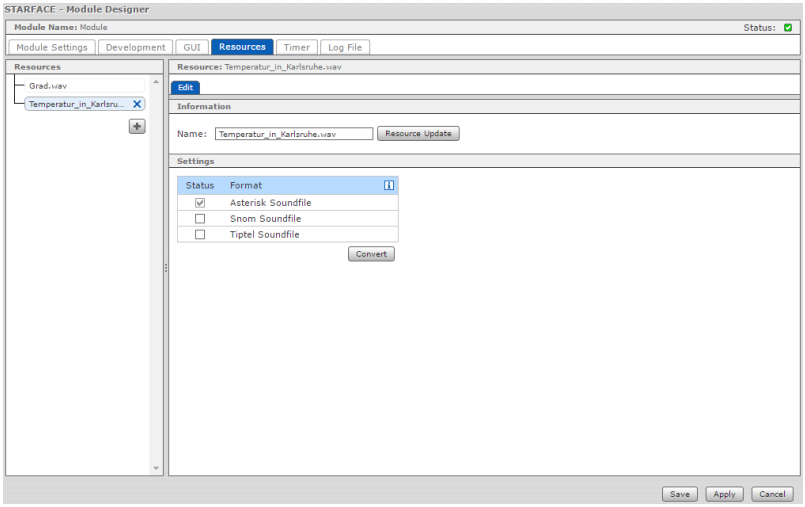
Anleitung zur Erstellung des Beispielmoduls

Zur Umsetzung unseres Beispieles verwenden wir zwei Audio-Dateien, die im Bereich Ressourcen in das Modul geladen werden müssen.

Hinweis: Einige der Komponenten findet ihr erst, wenn im Modul Designer der *Expert mode* im Bereich *Components* aktiviert ist.



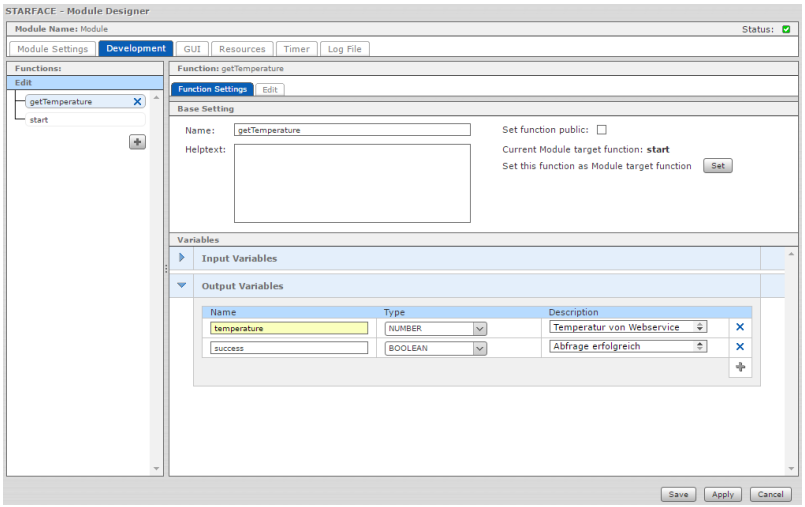
Festlegen der Grundeinstellungen, wie Name und Hersteller. Konfiguration als Modul vom Typ *Standard.*



Hochladen der Audiodateien für die Ansagen.

Für das Beispiel verwenden wir die Ansage "Die Temperatur in XXX beträgt" und "Grad". Diese können unter dem Reiter *Resources* hochgeladen werden.

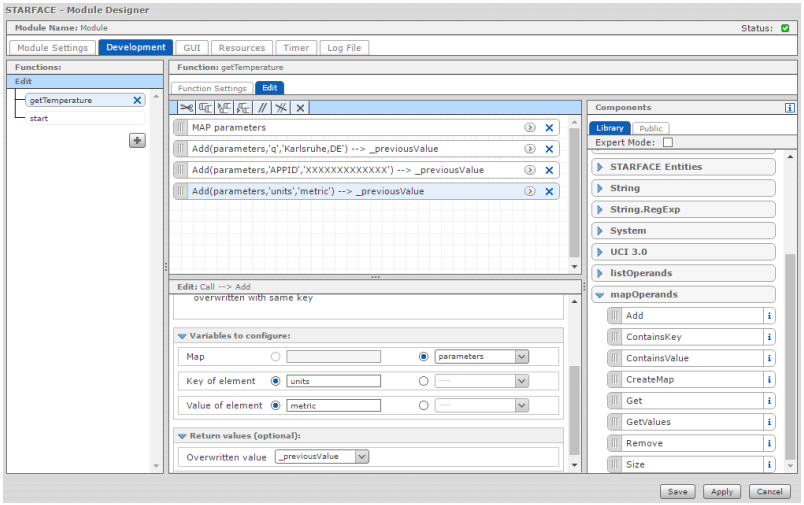
Wichtig: Nach dem Hochladen, müssen diese in das Format Asterisk Soundfile konvertiert werden.



Unter dem Bereich *Development* legen wir nun zwei neue Funktionen mit der Bezeichnung **start** und **getTemperature** an.

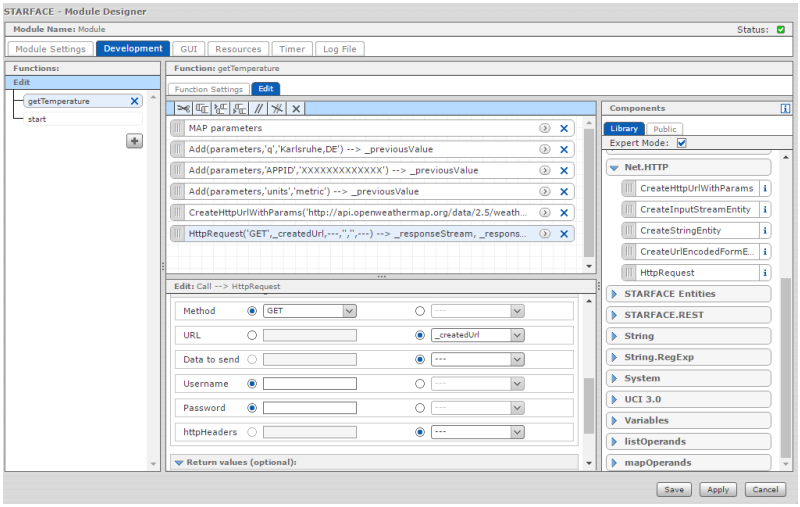
Der Funktion getTemperature fügen wir zwei Ausgabe Variablen (*Output Variables*) hinzu.

- **temperature** als Number
- **success** als Boolean



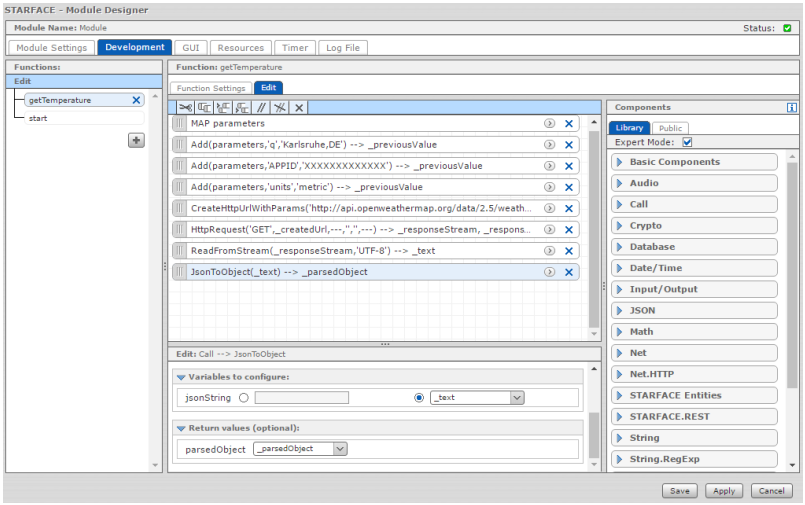
Im nächsten Schritt erstellen wir eine Variable **parameters** vom Typ *MAP*, in welcher wir die benötigten Parameter für die Webservice-Abfrage zusammenstellen.

Die Funktion *Add*, zum Hinzufügen von Elementen in eine *MAP*, findet sich im Bereich *mapOperands*.



Nun starten wir die Abfrage des Webservices, indem wir die Parameter zur URL hinzufügen. Dafür verwenden wir die Komponente *CreateHttpUrlWithParams* (Bereich *Net.HTTP*). Die Adresse des Webservice lautet http://api.openweathermap.org/data/2.5/weather

Die eigentliche Abfrage führen wir nun mit der Komponente *HttpRequest* durch.



Jetzt kommen wir zur Auswertung der Rückgabe.

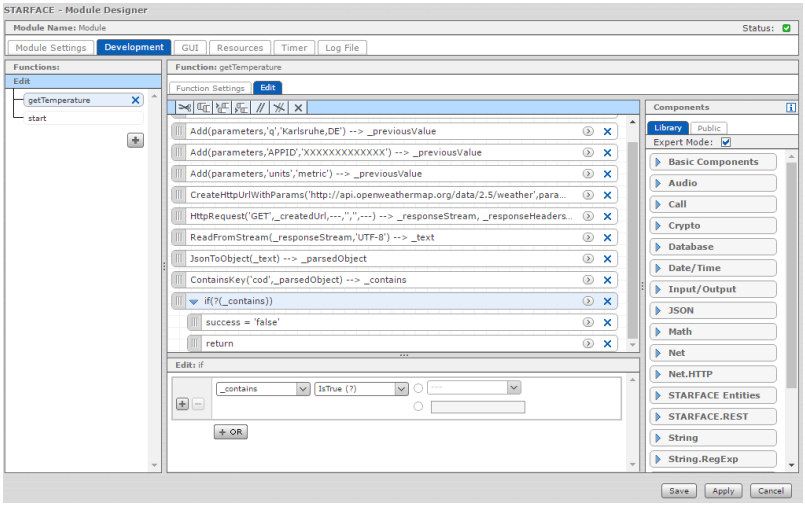
Zuerst wandeln wir den zurückgegebenen Stream in das Textformat mit *ReadFromStream* (Bereich *Input /Output*).

Danach parsen wir den JSON-Text in das STARFACE Objekt-Modell mit der Komponente *JsonToObject* (Bereich *JSON*).

Rückgabe:


```
{
        "coord": {
                "lon": 8.39,
                "lat": 49
        },
        "weather": [{
                        "id": 801,
                        "main":
"Clouds",

"description": "few clouds",
                        "icon": "02d"
                }
        ],
        "base": "stations",
        "main": {
                "temp": 0.23,
                "pressure": 1028,
                "humidity": 69,
                "temp_min": -1,
                "temp_max": 1
        },
        "visibility": 10000,
        "wind": {
                "speed": 5.7,
                "deg": 10
        },
        "clouds": {
                "all": 20
        },
        "dt": 1483618800,
        "sys": {
                "type": 1,
                "id": 4921,
                "message": 0.0058,
                "country": "DE",
                "sunrise": 1483600786,
                "sunset": 1483631089
        },
        "id": 2892794,
        "name": "Karlsruhe",
        "cod": 200
}
```

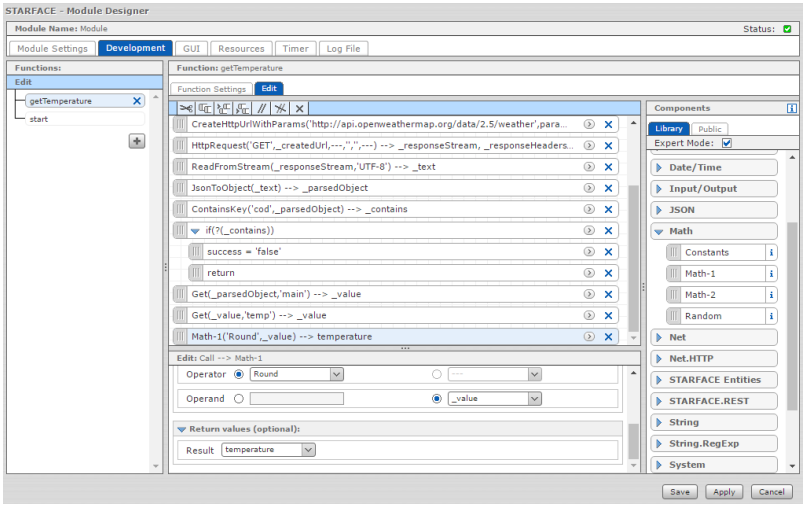


Prüfen ob Abfrage erfolgreich war.

Das zurückgegebene Objekt enthält eine MAP von Eigenschaften. Wenn die Abfrage erfolgreich war, enthält die MAP die Eigenschaft cod. Dies überprüfen wir nun mit den nächsten Schritten.

Zur Überprüfung verwenden wir folgende Komponenten:

- *ContainsKey (mapOperands):* Prüft, ob die *MAP* den *Key* **cod** enthält
- *if (Basic Components):* Bedingung, die die Rückgabe von *containsKey* erfolgreich war
- *variable definition (Basic Components):* hiermit setzen wir unsere Eingangsvariable **success** auf *false*, da die Abfrage fehlerhaft war.
- *return (Basic Components):* Bricht unsere Funktion **getTemperature** ab



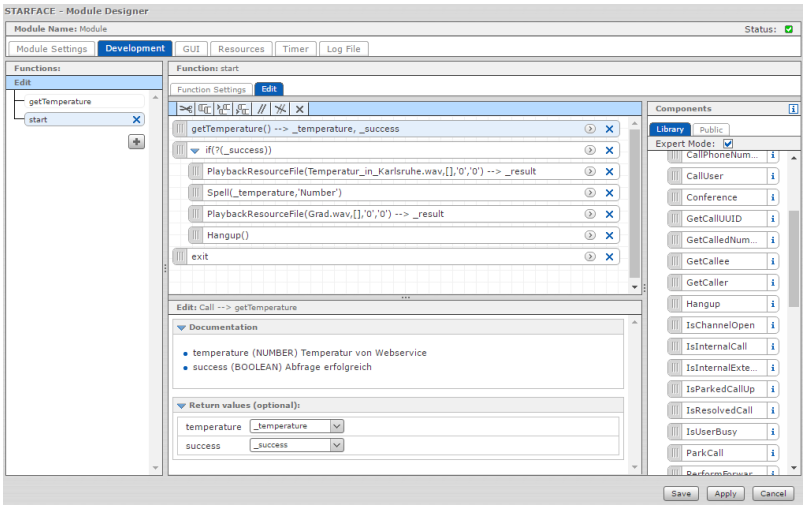
Temperaturwert extrahieren und runden.

Im JSON-Objekt findet sich die Temperatur im "Unterobjekt" **main**. Dort findet sich die Eigenschaft **te mp**. zum Extrahieren müssen wir somit zwei Maps auslesen.

Hierfür können wir die Komponente *Get (mapOperands)* verwenden.

Zuletzt nehmen wir den Wert und Runden ihn auf eine Ganzzahl. Die Funktion *Math-1 (Math)* bietet dazu die Option *Round* an. Das Ergebnis schreiben wir direkt in Ausgabevariable **temperature**.

Hiermit haben wir unsere eigene Komponente fertiggestellt, welche wir nun in unsere Funktion **start** verwenden können.

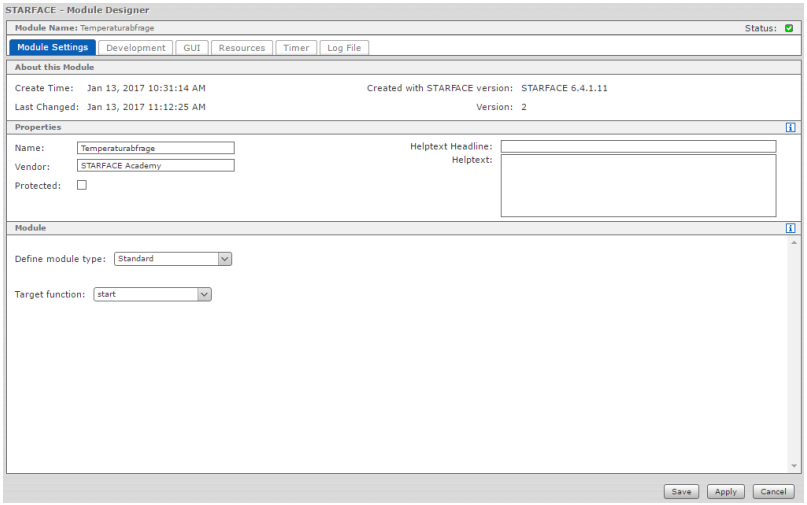


Hauptfunktion fertigstellen.

In unserer Funktion **start** fügen wir nun unsere eigene Komponente über den Reiter *Public* bei den Komponenten ein.

Mit folgenden Komponenten vervollständigen wir unser Modul:

- *if:* Bedingung, zum Prüfen, ob die Webservice-Abfrage erfolgreich war
- *PlaybackResourceFile:* Abspielen der ersten Ansage
- *Spell:* Ausgabe des Temperaturwertes
- *PlaybackResourceFile:* Abspielen des Wortes "Grad"
- *Hangup:* Anruf beenden
- *exit:* Modul beenden



Startfunktion auswälen.

im letzten Schritt, stellen wir sicher, dass unsere Funktion **start**, beim Anruf gestartet wird. Dazu gehen wir auf den Reiter *Module Setting* und wählen die Funktion bei *Target Function* aus.