Raccordement avec un portail métier (SSO)

Objectif de cette note : rappeler à l'éditeur d'un portail métier les standards SSO et plus précisément les impératifs pour se raccorder à un portail Publik. Ce texte pourra être inclus avec profit par les collectivités directement dans leur CCTP afin qu'il s'impose à l'éditeur métier.
Entr'ouvert ne fournit pas de base d'assistance à l'intégration ou à la spécification fonctionnelle d'un SSO. Si souhaité, cette assistance nécessitera un contrat de prestation auprès de la collectivité ou de l'éditeur/intégrateur du portail de l'éditeur métier.

Les fonctionnalités prévus lors d'un raccordement métier seront :

- authentification et déconnexion unique (SSO/SLO)
- appairage de comptes métiers avec les comptes du portail Publik via mise en place d'un pseudonyme
- remontée d'informations par des web-services porté par le portail métier interrogeable via le pseudonyme (uniquement si le SSO est en place)
  - demandes en cours
  - factures
  - autres (profil, inscriptions, etc...)

Authentification unique

Publik propose un seul mode de raccordement via le protocole OpenID Connect (OIDC), les spécifications à respecter sont :

- http://openid.net/specs/openid-connect-core-1_0.html (pour SSO), avec utilisation du flot "authorization code"
- https://openid.net/specs/openid-connect-rpinitiated-1_0.html uniquement pour la déconnexion (SLO) initiée depuis le portail métier
- http://openid.net/specs/openid-connect-frontchannel-1_0.html pour la déconnexion (SLO) initiée depuis Publik

Les spécifications désignées DOIVENT être respectées à la lettre, le nommage des attributs ("claim") et des scopes peut, par contre, être adapté.

Des implémentations de référence du protocole OpenID Connect pour différents langages et cadriciels sont disponibles sur https://openid.net/developers/libraries/

Le portail métier et ses éventuels web-services DOIVENT absolument être diffusés via le protocole HTTPS://.

Une authentification réussie fournit au portail un pseudonyme opaque appelé "sub" dans le jargon OpenID Connect, c'est une chaîne UTF-8 qui devra être conservé dans le schéma de donnée du portail et lié au compte du portail métier. La relation entre ce pseudonyme et les comptes du portail métier pourra être :

- 1-1 (1 compte publik pour 1 compte portail métier),
- 1-n (1 compte Publik vers plusieurs comptes portail métier),
- n-1 (plusieurs comptes Publik pour 1 compte portail métier)
- ou n-n (plusieurs comptes Publik pour plusieurs comptes portail métier).

Ce choix dépendra des usages et de la possibilité pour l'éditeur du portail de gérer les parcours de connexion nécessaire avec par exemple :

- une page de sélection du compte cible à la connexion après SSO,
- ou la possibilité pour les sessions applicatives du portail métier de donner accès à plusieurs comptes en même temps.

La processus de raccordement à des comptes existants (nommé aussi appairage), ou de création d'un nouveau compte suite à une authentification unique (SSO) réussie, relève de la seule responsabilité du portail métier mais on notera les cas suivants :

- Portail métier où les comptes sont purement déclaratifs (pur service en ligne) :
  - l'éditeur métier aura la possibilité de raccorder automatiquement le compte Publik à des comptes existant via l'adresse électronique ou de créer le compte automatiquement via les attributs servis au SSO, il sera de son ressort et de la collectivité de déterminer si c'est juridiquement valable, ou si cela nécessite l'envoi d'un courriel de validation.
- Portail métier avec des comptes métiers créés par ailleurs (compte famille, compte association par exemples). Le portail métier devra prévoir un moyen de lier un compte existant dans Publik à un comptes existant sur le portail métier, par exemples (liste non exhaustive) :
  - ajout sur toutes les factures de l'usager d'un identifiant et d'un mot de passe, qui seront demandés lors de l'appairage,
  - envoi d'un code ou d'un lien par SMS ou mail si un canal de communication validé existe déjà avec l'usager via son téléphone mobile ou son email,

- modération par des agents,
- raccordement basé sur l'adresse électronique validée sur Publik et sur le portail métier (si jugé suffisant juridiquement et sécuritairement).

Pré-requis

L'intégrateur du portail métier devra fournir à un administrateur de la plateforme Publik les informations suivantes :

- la liste des URLs de redirection après SSO : redirect_uris
- la liste des URLs de redirection après SLO initié par le portail métier : post_logout_redirect_uris
- l'URL unique de réception des SLO initié par la plateforme Publik : frontchannel_logout_uri

Attention : si les URLs de redirection ont des domaines différents et l'UUID n'est pas utilisé comme identifiant, penser à définir une URL de secteur.

L'administrateur de la plateforme Publik transmettra à l'intégrateur du portail métier :

- l'identifiant OIDC du portail : client_id
- le secret partagé pour authentification des appels web-service du portail à Publik : client_secret
- l'URL d'autorisation : authorization_endpoint_url (en général IDP_URL/idp/oidc/authorize/)
- l'URL du web-service de token : token_endpoint_url (en général IDP_URL/idp/oidc/token/)
- l'URL du web-service du profil utilisateur : userinfo_endpoint_url (en général IDP_URL/idp/oidc/user_info/)
  Une page explicitant cette configuration est disponible en ligne, en général IDP_URL/.well-known/openid-configuration)

Toutes les opérations de connexion ou déconnexion se font via des appels au fournisseur d'identité de la plateforme Publik de la collectivité, on désignera son URL par IDP_URL.

Description technique non-exhaustive de l'authentification unique

Une authentification unique débutera comme suit :
1. déclenchement de l'authentification sur le portail (appui sur un bouton, appel à une URL particulière, protection de toutes les pages du portail, c'est un point d'implémentation qui relève de l'éditeur du portail métier)

2. redirection du navigateur de l'utilisateur sur l'URL d'autorisation OpenID Connect du fournisseur d'identité de la plateforme Publik:

```
IDP_URL/idp/oidc/authorize/?client_id=...&...&redirect_uri=...&state=...&...
```

- le détail de l'appel se trouve dans les spécifications, on notera simplement ici les paramètres importants suivant:
  - client_id : identifie le portail métier auprès du fournisseur d'identité, la valeur de ce champ sera fournit par l'administrateur de la plateforme Publik
  - redirect_uri : URL sur laquelle l'usager sera re-dirigé après authentification, elle devra être déclarée auprès de l'administrateur de la plateforme Publik (plusieurs sont possibles, mais pas de wildcard)
  - state : paramètre opaque retourné au retour sur le portail métier si le SSO échoue ou réussi, il permet au portail de maintenir un état entre la requête de SSO et sa réponse (par exemple URL de retour précise sur le portail, sachant que l'URL de retour au SSO ne peut pas varier énormément)

3. le fournisseur d'identité de la plateforme Publik procède à l'authentification de l'usager, s'il n'est pas déjà authentifié, puis s'il n'a pas encore donné son autorisation lui présente un formulaire lui demandant de confirmer sa volonté de s'authentifier auprès du portail en fournissant un certain nombre d'attributs.

4. le fournisseur d'identité Publik redirige le navigateur de l'utilisateur sur l'URL de retour (ou de callback) déclaré par le portail métier par exemple,

```
https://portail-metier.collectivite.fr/oidc/callback/?code=xxx&state=yyy
```

5. le portail métier à la réception de cette requête doit :
5.1. éventuellement **utiliser le paramètre state** pour retrouver un état conservé (URL de retour sur le portail, identifiant métier de la personne demandé préalablement, jeton transmis par mail ou SMS permettant l'appairage, etc...) en session
5.2. **appeler le web-service token** (IDP_URL/ipd/oidc/token/) du fournisseur d'identité en s'authentifiant avec son client_id et son client_secret (fourni lui aussi par l'administrateur de la plateforme Publik) permettant d'échanger code contre un access_token ; l'access token est un jeton qui permettra d'appeler le web-service fournissant les informations sur l'utilisateur. Le web-service token fournit aussi le **sub** ou identifiant unique de l'utilisateur, c'est cet identifiant qui sert de clé à la liaison entre Publik et le portail-métier. Enfin, le web-service token fournit deux valeurs **iss** (identifiant de l'émetteur) et **sid** (identifiant de la session) qui doivent être enregistrés dans la session créée sur le portail métier, en vue d'assurer la mécanique de déconnexion (voir plus bas).

5.3. **appeler le web-service user-info** (IDP_URL/idp/oidc/user_info/) pour obtenir les attributs de l'utilisateur qui lui sont autorisés.

5.4. déterminer si le sub est connu

- si le sub est connu, il connecte l'usager à son compte ou un de ses comptes (voir plus haut)
- si le sub est inconnu, il doit lancer un processus d'appairage ou de création de compte, nécessitant des interaction ou pas avec l'utilisateur (cette partie est entièrement à la charge de l'éditeur métier)

Description technique non-exhaustive de la déconnexion unique

La déconnexion pourra avoir lieu :

- sur la plateforme Publik, et dans ce cas le portail métier en sera notifié si un SSO a eu lieu vers lui durant la session courante en suivant la spécification "frontchannel logout"
- sur le portail métier, et dans ce cas il DOIT le notifier à la plateforme Publik en suivant la spécification "RP initiated logout"

SLO initié par Publik (Frontchannel Logout)

Voir http://openid.net/specs/openid-connect-frontchannel-1_0.html#RPLogout

La plateforme Publik procédera à une re-direction sur l'URL frontchannel_logout_uri via une iframe invisible, le portail métier devra procéder à la déconnexion de la session en cours sur toute requête à cette URL et faire en sorte qu'elle ne puisse jamais être mise en cache par le navigateur.

Deux paramètres sont envoyés dans la query-string de l'URL frontchannel_logout_uri : iss et sid. Ce sont ceux qui ont été reçus lors de l'appel au web-service token au début de la création de la session (voir plus haut). Ces deux paramètres permettent de retrouver la session en jeu sur le portail métier, et de la détruire afin d'assurer la déconnexion.

Note : il convient de régler les entêtes de sécurité de l'URL pour que celle-ci soit utilisable dans une iframe. Notamment il ne faut pas renvoyer d'entête de restriction X-Frame-Options, ni de limitation via Content-Security-Policy.

SLO inité par le portail métier (RP initiated Logout)

Voir https://openid.net/specs/openid-connect-rpinitiated-1_0.html

Le portail-métier devra rediriger le navigateur de l'utilisateur sur l'URL IDP_URL/idp/oidc/logout/, il pourra passer une URL de retour via le paramètre post_logout_redirect_uri. Cette URL devra être préalablement déclarée auprès de l'administrateur Publik.

Remontée d'information

Publik propose un point d'entrée unique pour la gestion de la relation usager, à ce titre il est important que Publik puisse centraliser toutes les informations importantes des portails métiers tiers. Pour cela nous proposons la possibilité d'appeler des web-services portés par les portails métiers en vue d'afficher des informations dans les pages du portail citoyen de Publik.

Pré-requis

- tous les web-services devront être servis via HTTPS:// et une authentification HTTP Basic
- tous les web-services devront échanger en utilisant le format JSON en entrée (POST) et en sortie
- en sortie toutes les réponses des web-services devront suivre le formalisme suivant :
  - dans le cas passant :

```
{
  "err": 0,
  "data": {}
}
```

    - err contiendra 0 si tout s'est bien passé, sinon 1 ou une chaîne de caractère indiquant le type de l'erreur,
    - data contiendra les données en sortie du web-service dans un dictionnaire JSON
  - dans le cas non passant :

```
{
  "err": "code-erreur-xxx",
  "err_desc": "",
}
```

    - err contiendra un code erreur (éventuellement 1 si rien de plus précis n'est décidé)

    - err_desc contiendra un texte décrivant l'erreur.
- les codes d'erreur HTTP (4xx, 5xx) ne devront être utilisé qu'en cas d'erreur exceptionnelle système (mauvaise URL invalide, erreur interne, etc..) jamais pour retourner une erreur prévue, ceci pour différencier les erreurs techniques des erreurs métiers.
- l'identifiant du compte/profil visé dans le portail ne devra se faire que via le sub mis en place via le SSO, ce sub pourra être passé soit dans le contenu JSON (POST en JSON, {"sub": "xxx"}), soit comme paramètre d'URL (https://portail/web-service/factures/?sub=xxxx)

Affichage de demandes en cours dans le portail-métier

Publik étant spécialisé dans la mise en place de télé-services, il y a un formalisme particulier à adopter pour faire remonter des demandes dans le portail Publik.

Ce téléservice devra répondre sur un GET et prendre en entrée le sub comme paramètre de l'URL (le chemin de l'URL n'est pas imposé). La réponse devra avoir la forme exacte suivante :

```
GET /api/demandes/?sub=xxx
```

```
{
  "err": 0,
  "data": [
     {
        "datetime": "2018-03-04 12:34:32",
        "name": "Demande de carte de stationnement",
        "status": "En attente d'information",
        "form_number": "1234",
        "form_status_is_endpoint": false,
        "url": "https://portail-metier/demandes/1234/",
        "draft": false
     }
  ]
}
```

Champ	Type	Obligatoire	Description
datetime	string, la date au format ISO8601 YYYY-MM-DD HH:MM:SS	Oui	la date de création de la demande
name	string	Oui	le nom du type de demande
form_number	string	Oui	l'identifiant de la demande (numéro ou autre)
url	string	Oui	lien profond vers la demande, si une URL par demande n'est pas disponible le portail métier peut se contenter de renvoyer l'URL d'une page affichant toutes les demandes en cours
status	string	Oui	le statut actuel de la demande, "Nouvelle", "En cours", "En attente d'information", à définir par le portail mais la chaîne doit être présentable directement à l'usager, ce ne doit pas être un code
form_status_is_endpoint	booléen	Non	vrai si le statut de la demande indique qu'elle est terminée
draft	booléen	Non	vrai si la demande est un brouillon

Affichage des factures en cours venant du portail-métier

Publik contient nativement un système de remontée et paiement de factures, pour permettre une intégration des factures exposées par un portail métier tiers, il y a là aussi un formalisme particulier à respecter.

Le portail métier DOIT fournir un web-services porté par une URL de base que nous nommerons BASE_URL, de la forme BASE_URL/invoices/, le dernier élément du chemin est obligatoire.

Ce téléservice devra répondre sur un GET et prendre en entrée le sub comme paramètre de l'URL (le chemin de l'URL n'est pas imposé). La réponse devra avoir la forme exacte suivante :

```
GET /api/invoices/?sub=xxx
```

```
{
   "err": 0,
   "data": [
              {
            "id": "939456",
            "label": "restauration août 2015",
            "amount": "37.26",
            "total_amount": "37.26",
            "created": "2015-08-01",
            "pay_limit_date": "2015-09-29",
            "paid": false,
            "no_online_payment_reason": "litigation",
            "payment_url": "https://portail-metier/factures/934395/pay/",
            "pdf_url": "https//portail-metier/factures/934395/pdf/F20180192.pdf",
             ... (autres informations si le logiciel backend en donne)
        },
   ]
}
```

Champ	Type	Obligatoire	Description
id	string, libre	Oui	identifiant interne au portail métier de la facture, uniquement pour debug
amount	string, décimal	Oui	montant du reste à payer, nombre décimal formaté avec un point décimal et non une virgule décimale (à l'anglaise)
total_amount	string, décimal	Oui	montant total de la facture, format identique à amount
created	string, date ISO	Oui	date de création de la facture, format ISO8601 YYYY-MM-DD
pay_limit_date	string, date ISO	Oui	date limite de paiement de la facture, date exlue (i.e. à partir de cette date le paiement n'est plus possible), format ISO8601 YYYY-MM-DD
payment_url	string, URL	Non	l'URL pour payer la facture si celle-ci est payable en ligne
pdf_url	string, URL	Non	l'URL vers le PDF de la facture si celle-ci est téléchargeable
non_online_payment_reason	sttring, enum	Non	si payment_url est absent, la raison éventuelle de la non-possibilité de paiement en ligne (contestation, prélèvement, délai dépassé), voir plus loin les valeurs possibles

Les valeurs actuellement possibles pour no_online_payement_reason sont :

Valeur	Description
litigation	la facture pose un problème
autobilling	la facture est payée par prélèvement automatique

past_due_date	il n'est plus possible de payer cette facture en ligne car un délai est dépassé

Remontée d'autres informations[1]

Pour tout autre type d'information nous imposons le formalisme générique suivant, en plus des pré-requis donnés plus haut :

- le web-service doit répondre sur un GET
- il doit accepter un seul paramètre à cette url: sub

Ce web-service retournera une description du formatage attendu pour ces donnés selon le formalisme suivant :

- data contiendra une liste de données qui seront transformés en HTML, chaque donnée est décrite par un dictionnaire et possède un type, ex:

```
{
   "type": "block",
   "label": "Mes informations familles",
   "content": [...],
   "edit_url": "https://portail-metier/ma-famille/",
}
```

Type	Description
block	un bloc composé d'autres éléments, permet de grouper
text	un paragraphe, peut contenir un texte simple, pré-formaté ou formaté en HTML, en utilisant uniquement les balises autorisés dans un noeud <p> du HTML (balises inline, , <i/>, <b/>), ainsi qu'un lien d'édition (pictogramme stylo dans le coin haut-droit du contenu)
table	un tableau, contenant des lignes qui contiennent soit des entête header soit des cellules de contenu text

Tous les types acceptent les attributs suivants :

- label qui ajoutera un titre avec le bon niveau d'imbrication avant le contenu,
- id et class (respectivement une chaîne et une liste de chaînes) permettant de donner des valeurs pour les attributs HTML correspondant, simplifiant l'application d'une feuille de style.

Exemple de présentation d'un profil famille venant d'un portail famille :

```
{
  "err": 0,
  "data": {
     "type": "block",
     "label": "Ma famille",
     "edit_url": "https://portail-famille/ma-famille/edit/",
     "content": [
         {
            "type": "text",
            "id": "adresse",
            "label": "Adresse",
            "pre": true,
            "content": "1 rue du calvaire\nXX100 MAVILLE"
         },
         {
            "type": "text",
            "id": "parent1",
            "class": "parent",
            "label": "Premier parent",
            "html": true,
            "content": "Jean-Michel <b>DUPOND</b>, né le 12 décembre 1964 à Marseille",
```

```
        },
        {
            "type": "text",
            "id": "parent2",
            "class": "parent",
            "label": "Second parent",
            "html": true,
            "content": "Régine <b>DUPOND</b>, né MARTIN le 12 décembre 1964 à Lyon"
        }
        {
            "type": "block",
            "label": "Enfants",
            "content": [
                {
                    "type": "text",
                    "content": "Kévin DUPOND, 5 ans, né le 22 mars 2013",
                }
            ]
        }
    ]
  }
}
```

Le but de ce formalisme est de laisser à la fois un peu de liberté au portail métier sur les contenus et aussi la liberté pour Publik d'appliquer des feuilles de style aux contenus.

Coté Publik, ces contenus seront présentés via des cellules JSON (cellule associant un web-service JSON avec un template).

[1] technologie équivalente https://github.com/portabletext/portabletext

Dépôt d'une demande

Voir https://doc-publik.entrouvert.com/dev/echanges-logiciel-de-demandes-metier/ en sachant que l'information « sub » peut être envoyée dans la demande si elle doit être reliée à un compte dans le logiciel cible.

Fichiers

sso-et-appairage.pdf	499 ko 05 novembre 2019	Marie Kuntz