TraceConnect

V1.0

Documentation version 1.0.0

Contents

1. Introduction

Le TraceConnect ou TConnect est un ensemble de packages qui rendre facile pour les développeurs d'intégrer leurs applications avec la base de traces ainsi que le serveur d'assistance.

Le TraceConnect comprend trois modules :

- tService
  - API pour le stockage et l'accès aux traces modélisées en JavaScript.
- tApp
  - API pour les requêtes non-trace nécessaires pour maintenir le lien entre l'application end-user et les autres systèmes.
- tAssistance
  - API pour les requêtes de haut niveau sur les traces (transformation, la visualisation, l'analyse, l'exploitation).

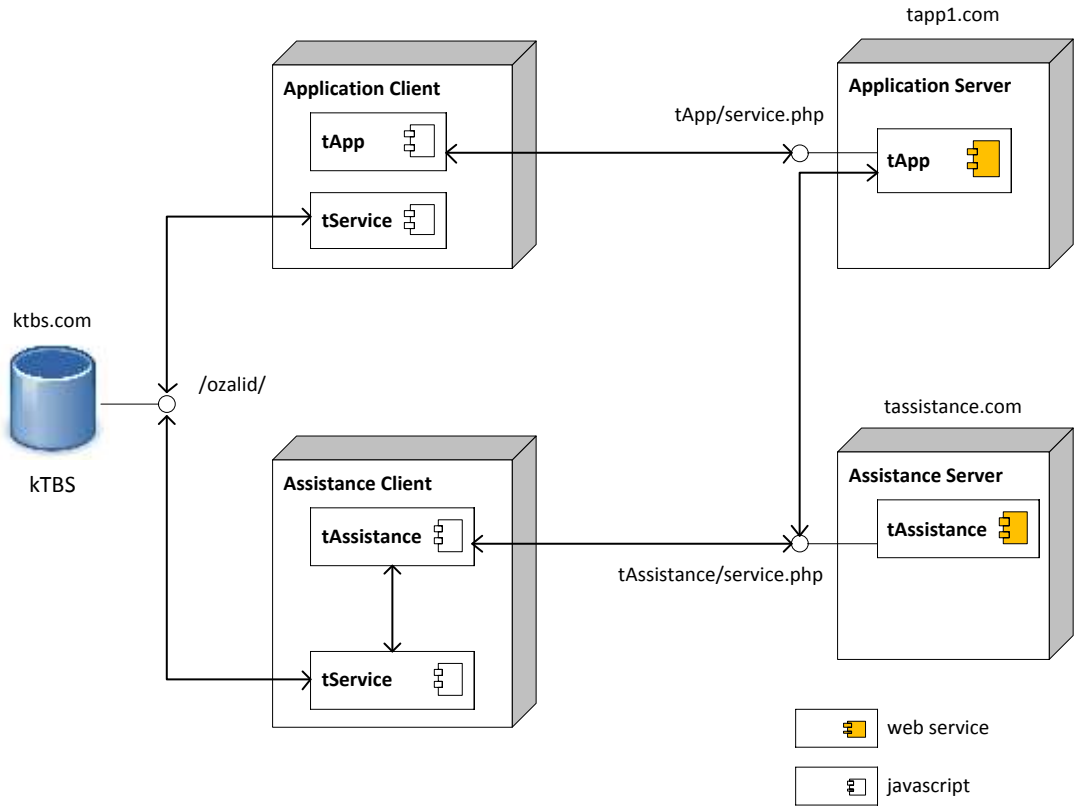

Figure 1. Schéma communication

Figure 2 Schéma illustrant l'usage des API sur l'environnement OZALID : l'application cliente utilise l'API tService pour enregistrer sur un SGBT (le kTBS du Liris dans le cas précis) les observés collectés; Elle délègue à un assistant générique le soin de visualiser, transformer, naviguer… l'expérience ainsi tracée. L'assistant générique illustre les possibilités de l'API tAssistance, tout en utilisant l'API tService pour les requêtes au SGBT (ici le kTBS).

2. Prise en main

2.1. Installation & tutoriaux

Pour installer TConnect:

https://github.com/ahle/tconnect

2.2. Enregistrer et collecter les éléments observés

Après le KTBS et le service d'assistance en ligne, vous pourriez enregistrer les éléments observés. Pour faire cela, vous devez déclarer le package tService dans la tête de toutes les pages de l'application que vous voudriez collecter les traces.

```
01 <html>
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
04 <title>Application 1</title>
05 <link href="css/trace.css" rel="stylesheet" >
06 <script type="text/javascript" src="your_path_tService/js/js.php"></script>
07
08 </head>
09 <body>
10 …
11 <div>
```

Ensuite, vous pourriez utiliser le package tService afin de créer une nouvelle trace et de collecter les éléments observés (obsels). Par exemple :

```
01 <div>
02 Demo buttons
03 <button id="bta" onclick="click_a()">A</button>
04 <button id="bta">B</button>
05 <button id="bta">C</button>
06 </div>
07 <script type="text/javascript">
08 function click_a(){
09
10  /*tService.trace_open({
11       ktbs_base: "http://your.server.name/ktbs/your_base/",
12       name: "trc_demo",
13       success: function(){console.log("success is callbacked");},
14       error: function(jqXHR,textStatus, errorThrown){console.log("error is
callbacked.");}
15       });*/
16  var begin = end = (new Date()).getTime();
17    tService.trace_put_obsels({
18       trace_uri: "http://your.server.name/ktbs/your_base/trc_demo/",
19       model_uri: " http://your.server.name/ktbs/your_base/model_demo/",
20       obsel: {
21           type: "click",
```

```
22            begin: begin,
23            end: end,
24            subject: "user_demo",
25            application: "app1"
26        },
27        success: function(){console.log("success is callbacked");},
28        error: function(jqXHR,textStatus, errorThrown){console.log("error is
callbacked.");}
29    });
30 }
```

2.3. Authentification et Synchronisation session utilisateur

Le serveur assistance est une application web autonome mais il n'a pas d'une forme authentification propre. A tout moment, l'utilisateur application est aussi l'utilisateur assistance. Pour faire cela, une cryptographie symétrique (clé secrète) et un protocole de synchronisation session sont utilisé. Il y a trois scénarios dans le protocole d'authentification et synchronisation session utilisateur :

1. Partager la clé secrète
2. Créer une nouvelle session utilisateur
3. Synchronisation utilisateur

2.3.1. Partager la clé secrète

Pour sécuriser les messages échangés entre le serveur application et le serveur assistance, une clé secrète est partagée entre deux serveurs au travers d'un protocole sécurisé (HTTPS) (Figure 3).

Cette clé est unique et privée avec les autres serveurs. Le serveur assistance permet à plusieurs serveurs application de se connecter à lui. Donc, un ensemble de clés sont gérées par le distributeur de clés.

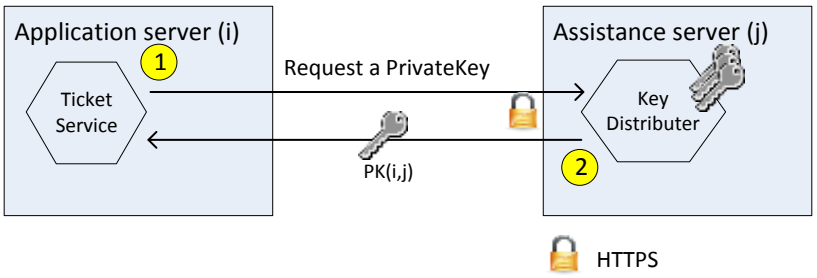

Figure 3. Partager la clé secrète

2.3.2. Créer une nouvelle session utilisateur

En fait, les applications permettent typiquement aux utilisateurs d'ouvrir une session web avec leurs systèmes assistance. Pourtant, ces systèmes assistance ne peuvent pas identifier quel utilisateur qui est en train de travailler avec le système certainement. Donc, l'information de la session utilisateur en cours est encryptée et envoyée au serveur assistance comme le schéma ci-dessous (Figure 4).

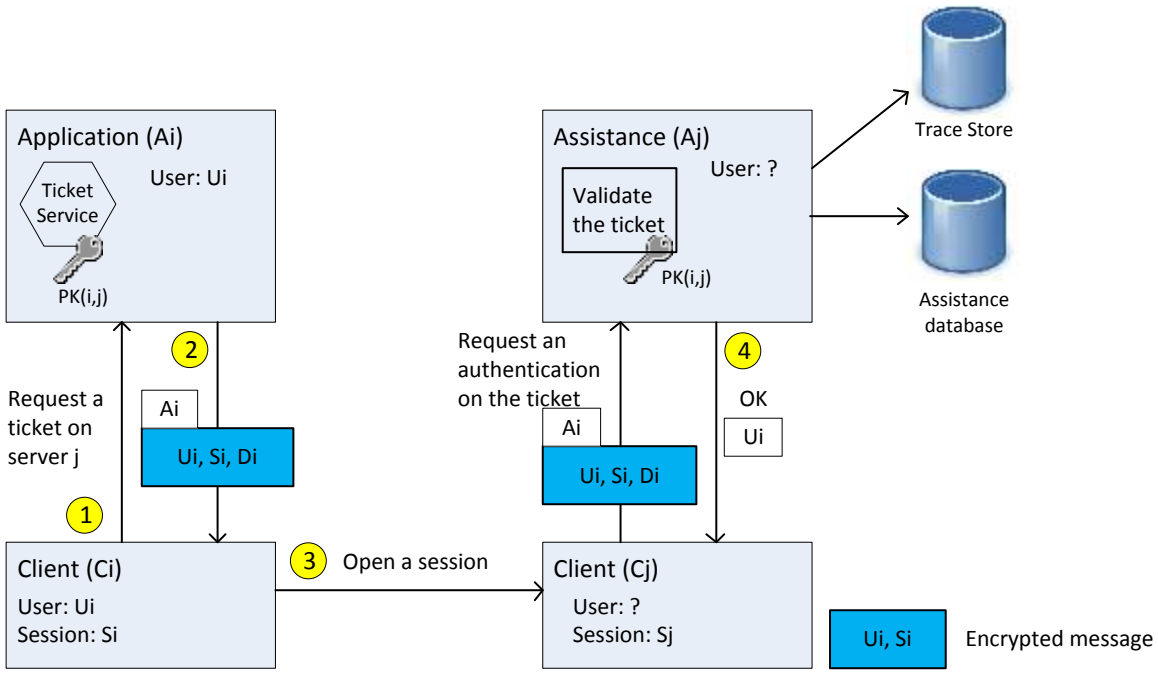

Figure 4. Créer une nouvelle session utilisateur

Avant l'ouverture d'une nouvelle interface assistance, une requête est envoyée au serveur application afin d'obtenir un ticket encrypté par la clé secrète partagée (1). Ce ticket contient l'information sur la session utilisateur courant comme identifiant utilisateur, identifiant session, etc. (2)

Immédiatement après l'ouverture d'une nouvelle session, ce ticket est envoyé au serveur assistance pour le valider (3). Ensuite, le serveur assistance décrypte le ticket avec la clé secrète correspondant l'application. Si le décryptage est succès avec la clé secrète, l'identifiant utilisateur de l'application est utilisé comme l'identifiant utilisateur sur le serveur assistance (4).

Le temps où la requête de ticket est reçue (Di) est ajouté dans le message pour limiter la réutilisation de ticket ainsi que la possibilité de décryptage des attaqueurs.

2.3.3. Synchronisation utilisateur

En réel, l'état de session application, surtout l'état utilisateur, peut être changé par l'utilisateur (i.e. changement d'utilisateur, connexion, déconnexion) (1). Ces changements sont propagés au serveur assistance automatiquement pour assurer la synchronisation entre deux sessions utilisateur sur deux serveurs différents (2).

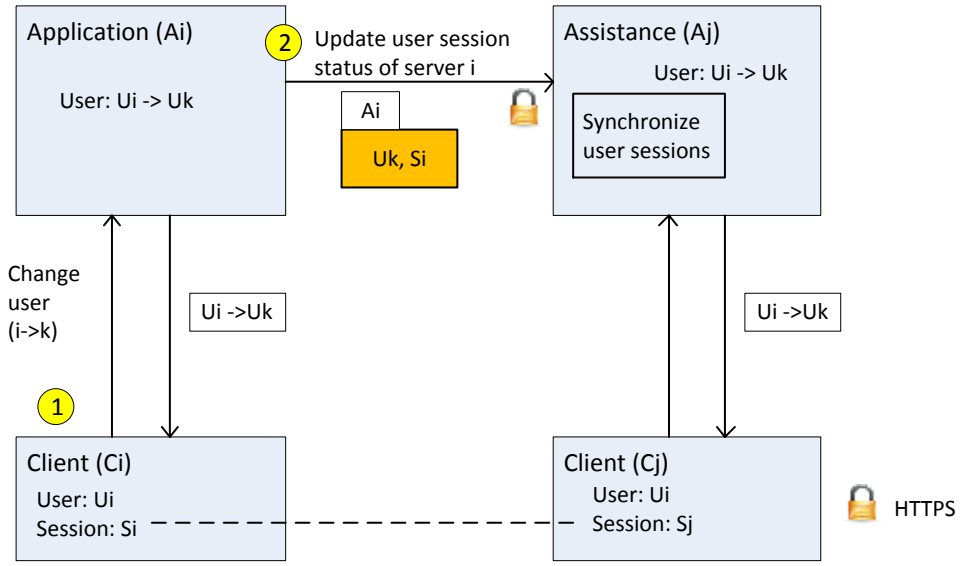

Figure 5. Synchronisation utilisateur