TraceConnect

V1.0

Documentation version 1.0.0

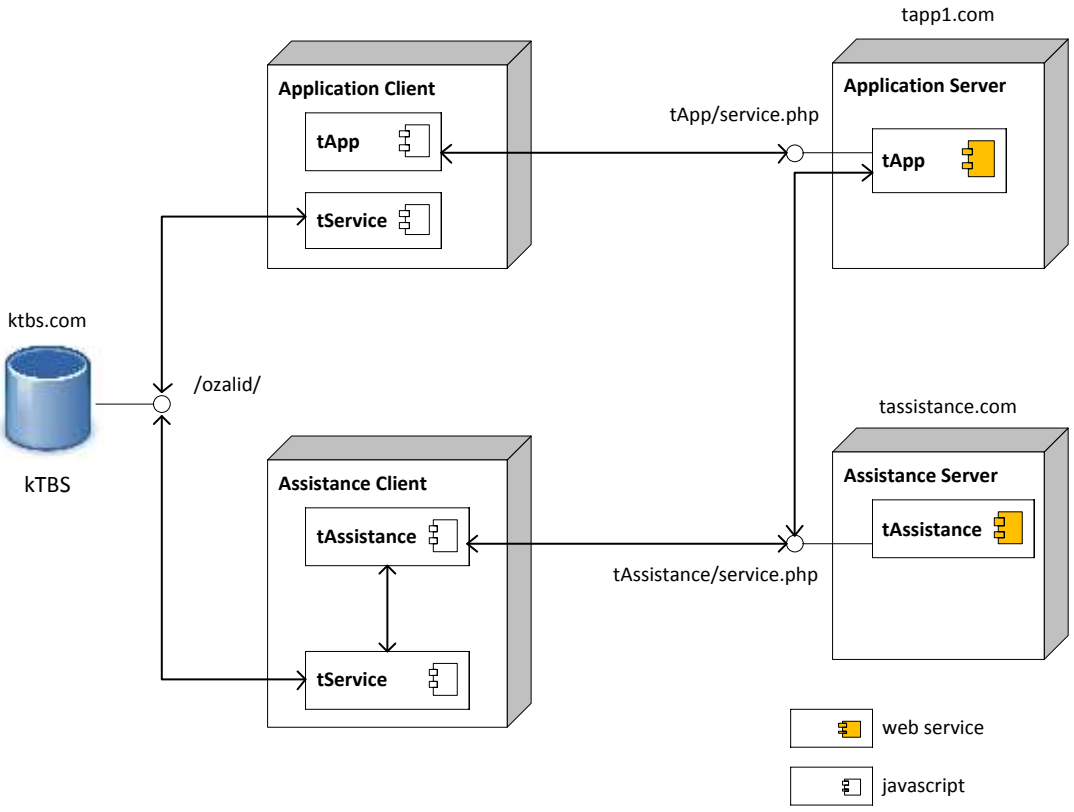
Contents

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).



2. Prise en main

2.1. Déploiement TConnect

Le paradigme « assistance à base de trace » comprend trois composants Application – SGBT (système de gestion de base de traces) – Assistance. Ces composants peuvent avec les composants de TConnect.

2.2. Installation et exécution KTBS

Premièrement, vous devez avoir une base de traces en ligne. La base de traces par défaut est le KTBS (un système à base de traces). Pour installer et exécuter le KTBS, vous suivrez la guide https://kernel-for-trace-based-systems.readthedocs.org/en/latest/tutorials/install.html

Après la configuration et l'exécution, vous auriez l'URL de votre KTBS ressemblant à http://your.server.name/ktbs/.

2.3. Installation le service d'assistance

Deuxièmement, vous voudriez avoir un service d'assistance en ligne. Le package Assistance (tAssistance) est écrit en PHP et JavaScript. Donc, vous devez avoir un serveur en ligne avec le mode PHP5 activé. Ensuite, vous mettez tout le package tAssistance dans le dossier web root (par exemple : sous Ubuntu « /var/www/ ») et activez le dossier tAssistance comme un nouveau site.

Après la configuration et l'exécution, vous auriez l'URL de votre service d'assistance ressemblant à http://your.assistance-server.name/tAssistance/.

2.4. 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 }
```

2.5. 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

2.5.1. 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 1).

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.

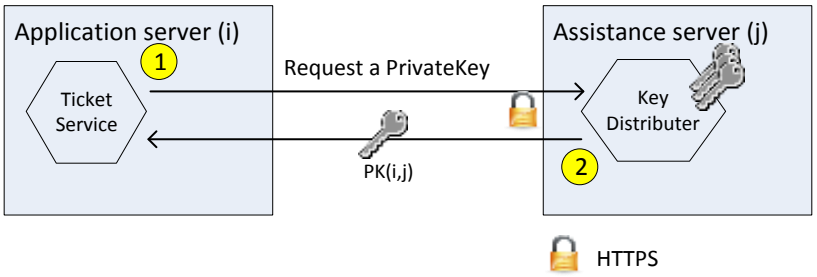


Figure 1. Partager la clé secrète

2.5.2. 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 2).

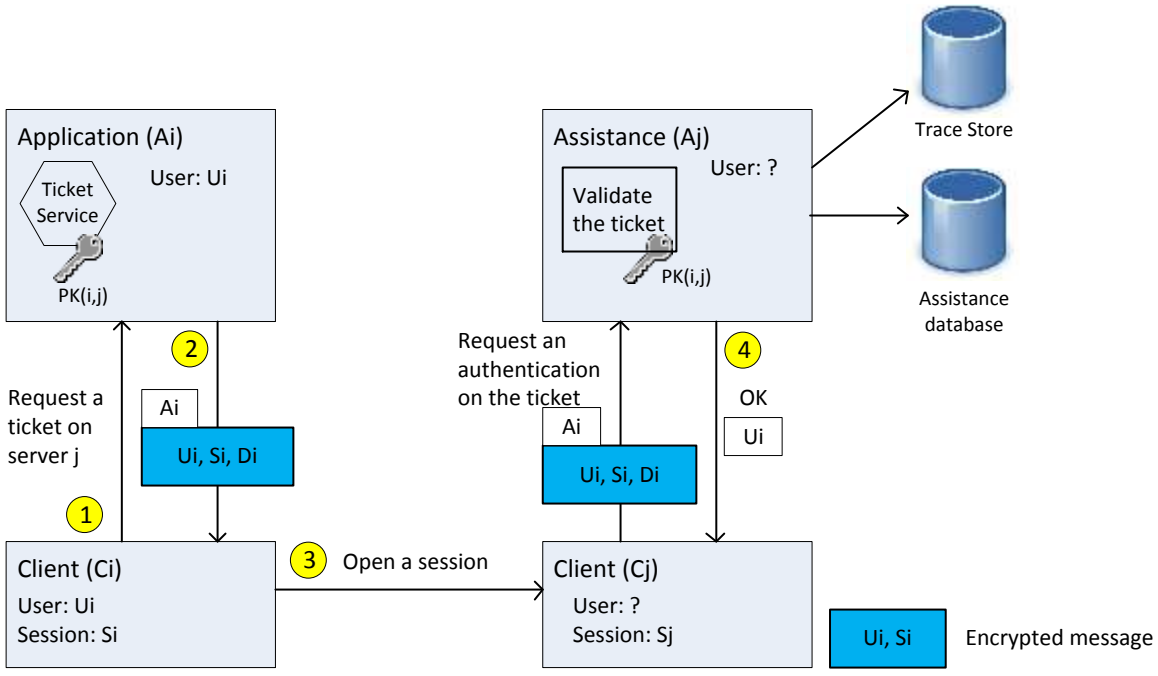


Figure 2. Créer une nouvelle session utilisateur

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.

2.5.3. 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).

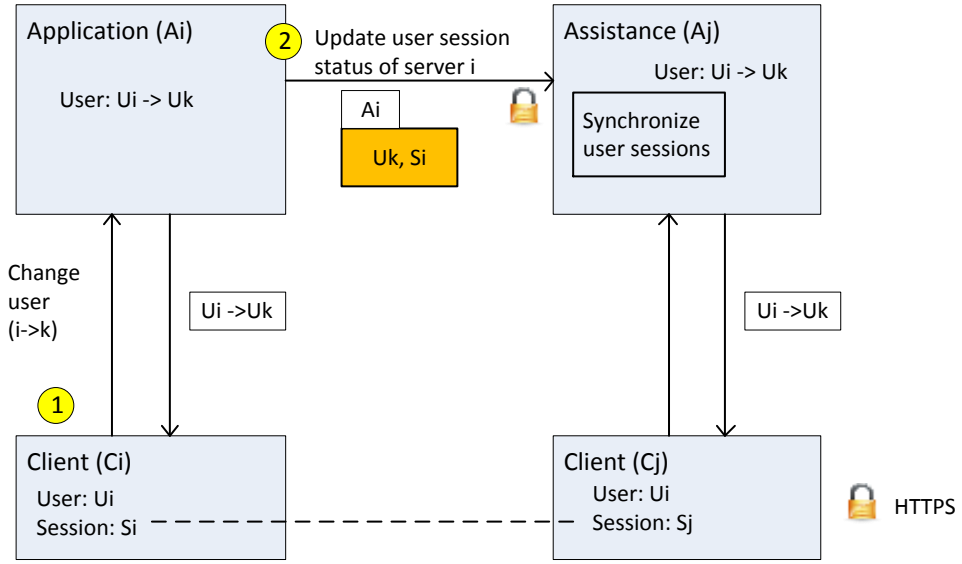


Figure 3. Synchronisation utilisateur

3. Références API

3.1. Trace Service (tService)

3.1.1. trace_open (options)

Cette fonction sert à créer une trace dans la base de trace.

Paramètre « options » (type: Object) est un ensemble de paires clé/valeur qui configurent la fonction.

- ktbs_base : L'URI de la base de traces
- name : le nom de trace
- success : la fonction callback qui s'est invoqué dans le cas la création de la trace est succès.
    - ret : le string de réponse de la base de données
- error : la fonction callback qui s'est invoqué dans le cas la création de la trace n'est pas réussie.
    - jqXHR : (jQuery XMLHttpRequest) remplace l'objet XMLHttpRequest native du navigateur
    - textStatus : un string décrit le type d'erreur qui se produit
    - errorThrown : un objet exceptionnel s'il se produit
- async (defaut=true) : Par défaut, toutes les requêtes sont envoyées de manière asynchrone. Si vous avez besoin requêtes synchrones, définir cette option à false.

3.1.2. trace_put_obsels (options)

Cette fonction sert à insérer les obsels dans une trace donnée.

Paramètre « options » (type: Object) est un ensemble de paires clé/valeur qui configurent la fonction.

- trace_uri : le chemin d'accès de la trace
- model_uri : le chemin d'accès du modèle des obsels insérés
- obsels : un ensemble d'objets représentant des obsels qui vont être mis dans la trace.
- success : la fonction callback qui s'est invoqué dans le cas la création de la trace est succès.
    - ret : le string de réponse de la base de données
- error : la fonction callback qui s'est invoqué dans le cas la création de la trace n'est pas réussie.
    - jqXHR : (jQuery XMLHttpRequest) remplace l'objet XMLHttpRequest native du navigateur
    - textStatus : un string décrit le type d'erreur qui se produit
    - errorThrown : un objet exceptionnel s'il se produit
- async (defaut=true) : Par défaut, toutes les requêtes sont envoyées de manière asynchrone. Si vous avez besoin requêtes synchrones, définir cette option à false.

3.1.3. trace_get_obsels(options)

Cette fonction sert à récupérer les obsels dans la trace.
Paramètre « options » (type: Object) est un ensemble de paires clé/valeur qui configurent la fonction.

- trace_uri : le chemin d'accès de la trace
- success : la fonction callback qui s'est invoqué dans le cas la récupération de la trace est succès.
    - obsels : l'ensemble des obsels récupérés.
- error : la fonction callback qui s'est invoqué dans le cas la création de la trace n'est pas réussie.
    - jqXHR : (jQuery XMLHttpRequest) remplace l'objet XMLHttpRequest native du navigateur
    - textStatus : un string décrit le type d'erreur qui se produit
    - errorThrown : un objet exceptionnel s'il se produit

3.2. Assistance Service (tAssistance)

3.2.1. obsel_serialize_html (options)

Cette fonction sert à sérialiser un obsel sous format HTML avec ses propriétés.
Paramètre « options » (type: Object) est un ensemble de paires clé/valeur qui configurent la fonction.

- obsel (type : Object) : l'obsel va se sérialiser.

La *valeur de retour* est un string représentant l'obsel en HTML.

3.2.2. draw_obsels (options)

Cette fonction sert à créer une vue interactive des obsels.
Paramètre « options » (type: Object) est un ensemble de paires clé/valeur qui configurent la fonction.

- parentNode (type : String ou DOMDocument): réfèrent à la position que la vue va être injectée
- obsels (type : ArrayObject) : un ensemble d'objets (obsels)
- getx (type : fonction) : la fonction récupère les valeurs représentant sur l'axe « x » de la vue.
- gety (type : fonction) : la fonction récupère les valeurs représentant sur l'axe « y » de la vue.
- width (Integer): la largeur de la vue
- height (Integer): la hauteur de la vue
- obselStyle : un script Javascript pour décorer l'obsel en SVG basé sur l'information de l'obsel

3.3. Assistance Server

3.3.1. User

3.3.1.1. Mettre à jour l'état de session utilisateur

Pour mettre à jour l'état de session utilisateur, créez une requête HTTP GET à l'URI du service assistance :

/srv/service.php

GET paramètres

Paramètre	Valeur	Description
App_id	Obligatoire	L'identifiant de l'application qui crée la requête
Session_id	Obligatoire	L'identifiant de la session utilisateur sur le serveur application
User_id	Obligatoire	L'identifiant de l'utilisateur en cours dans la session. Le paramètre est 0 si l'utilisateur s'est déjà déconnecté.

Réponse

Après l'envoi de la requête, l'application va recevoir une réponse HTTP.

Statut	Réponse/*Exemple*	Description
200	NULL	Le changement de l'état de session utilisateur est mis à jour
500	*Cannot connect to database*	Une erreur s'est produite

3.3.1.2. Obtenir une clé sécrète

Pour obtenir une clé sécrète sur le serveur application, créez une requête HTTP GET à l'URI du service assistance :

/srv/service.php

GET paramètres

Paramètre	Valeur	Description
App_id	Obligatoire	L'identifiant de l'application qui crée la requête

Réponse

Statut	Réponse/*Exemple*	Description
200	*fruoskfhd*	La clé en texte brut
500	*Cannot connect to database*	Une erreur s'est produite

3.3.2. Style

3.3.2.1. Créer un style

3.4. Trace extension for application (client) : tApp/service.js

3.4.1. Ouvrir une nouvelle fenêtre assistance

```
tApp.openAssistWindow() -> void
```

Description : ouvrir une fenêtre connectée au serveur assistance.

3.5. Trace extension for application (server) : tApp/service.php

3.5.1. Synchronisation session utilisateur

3.5.1.1. Obtenir une clé avec le serveur assistance

Pour obtenir la première clé sécrète avec le serveur assistance, créez une requête HTTP GET à l'URI de l'interface extension trace de l'application :

/tApp/service.php

GET paramètres

Paramètre	Valeur	Description
action	get_private_key	Obligatoire

Réponse

Statut	Réponse/*Exemple*	Description
200	NULL	L'application obtient une nouvelle clé avec le serveur assistance
500	*Cannot connect to database…*	Une erreur s'est produite

3.5.1.2. Obtenir un ticket valable sur le serveur assistance

Pour obtenir un ticket valable sur le serveur assistance, créez une requête HTTP POST à l'URI de l'interface extension trace de l'application :

/tApp/service.php

POST paramètres

Paramètre	Valeur	Description
action	get_ticket	Obligatoire

Réponse

Statut	Réponse/*Exemple*	Description
200	*H8RAqjwEepXVa7pzFd4jixFKDw6M1qZe6/oE+L0KFLRcpZlytQguuPEt+itzHXHnygd27QlTDHboWLCUW47Ot5HVxVejo9W0Ep60omQa3Wk=*	Le ticket encrypté par la clé sécrète entre l'application et l'assistance (le standard d'encryptage est AES-256-CBC). Voir http://www.openssl.org/docs/apps/enc.html#SUPPORTED_CIPHERS
500	*Cannot connect to database*	Une erreur s'est produite