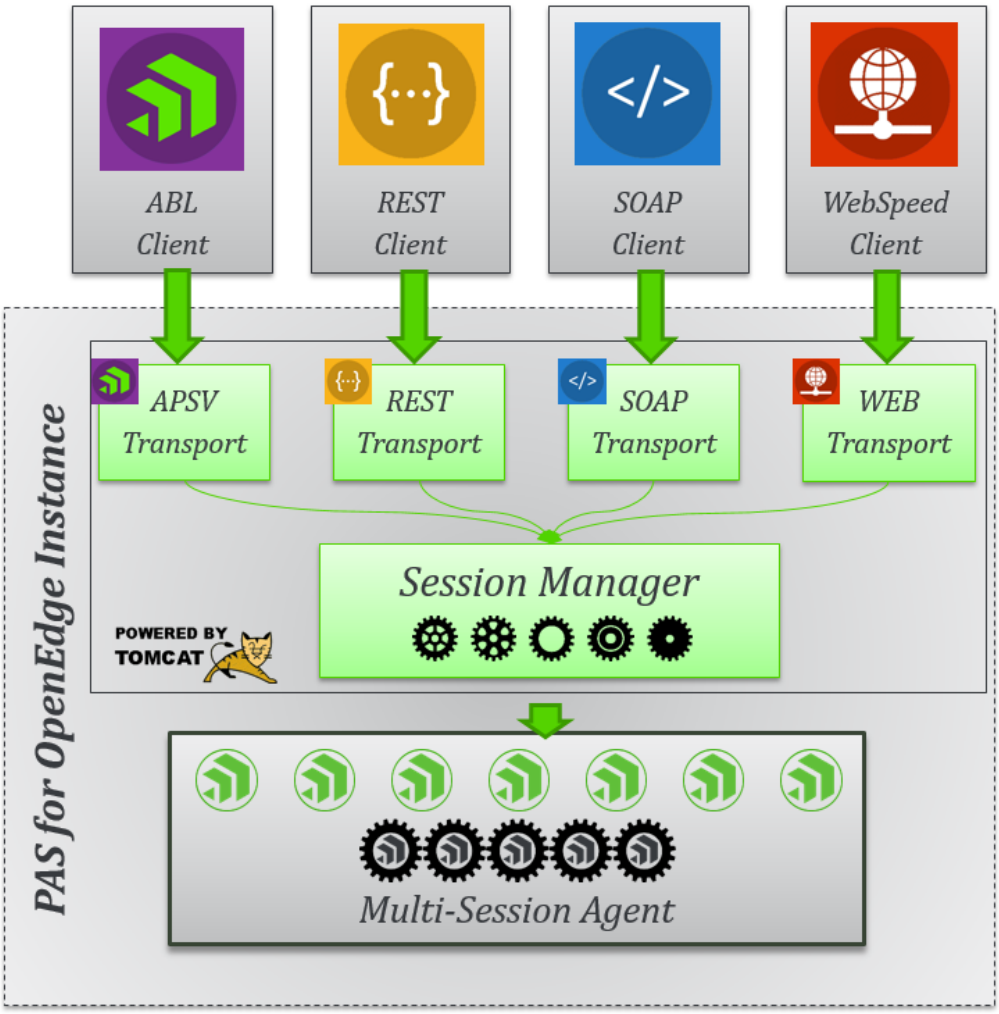
Appserver - PASOE

Serveurs d'Applications Progress

Ce qu'il faut Savoir



Créateur : Laurent KIEFFER

Version : 1.1

Création Document	20/05/2022	1.0
Mise à jour Document	23/05/2022	1.0
PASOE et Sécurité	14/06/2022	1.0
PASOE et Load Balancing Loadmaster	24/06/2022	1.1
PASOE et Docker	27/06/2022	1.1
PASOE et le Cloud	28/06/2022	1.1
PASOE en Conclusion	28/06/2022	1.1

Contents

Objectif de ce document

Rassembler des informations sur les apports du nouveau serveur d'application PASOE et constituer un recueil d'informations collectées suite à des retours d'expériences. Le document est donc amené à évoluer dans le temps.

Les informations en détail sur PASOE sont à revoir dans la documentation fournie par Progress software.

Introduction - Historique

La notion de serveur d'application existe depuis longtemps dans l'environnement Progress OpenEdge sous le nom AppServer depuis les versions Progress V9.

Cet Appserver avait comme seul objectif d'exécuter des traitements L4G (ABL) dans le cadre d'une application N-TIER.

Un traitement L4G développé en mode caractère ou client/serveur , sans lien avec l'IHM peut s'exécuter directement sur l'Appserver (en passant des paramètres).

Une application N-TIER pouvait être de plusieurs types :

- Frontal Windows application Progress accédant à du traitement métier sur une serveur d'application OpenEdge Appserver
- Frontal .NET ou Java (Openclient) accédant à du traitement métier sur une serveur d'application OpenEdge Appserver
- Application Web avec un traitement métier sur une serveur d'application OpenEdge Appserver (connu sous le nom Webspeed)

Au moment de l'introduction de la notion de Webservices SOAP, l'exposition de services pour les applications Progress a été mis en oeuvre en utilisant des serveurs de type JSE (Java Servlet Engine) afin d'exposer des services générés à partir de traitements Progress.

La notion de serveur JSE a été maintenu pour l'exposition de Services REST s'appuyant sur le serveur d'application historique Appserver.

L'Appserver a été largement utilisé par les clients et partenaires Progress avec plusieurs besoins :

- Répartir partiellement de la logique métier sur un serveur pour des besoins de performances et améliorer l'architecture Client/serveur
- Déplacer totalement la logique métier sur un serveur d'application pour découpler l'IHM du traitement.
- Exposer des Services SOAP/REST pour de l'intégration avec d'autres systèmes.

Introduction – Serveur d'Applications PASOE

Le serveur d'application PASOE (Progress Application for OpenEdge est apparu avec OpenEdge 11.5 en 2015.

Ses objectifs principaux on été :

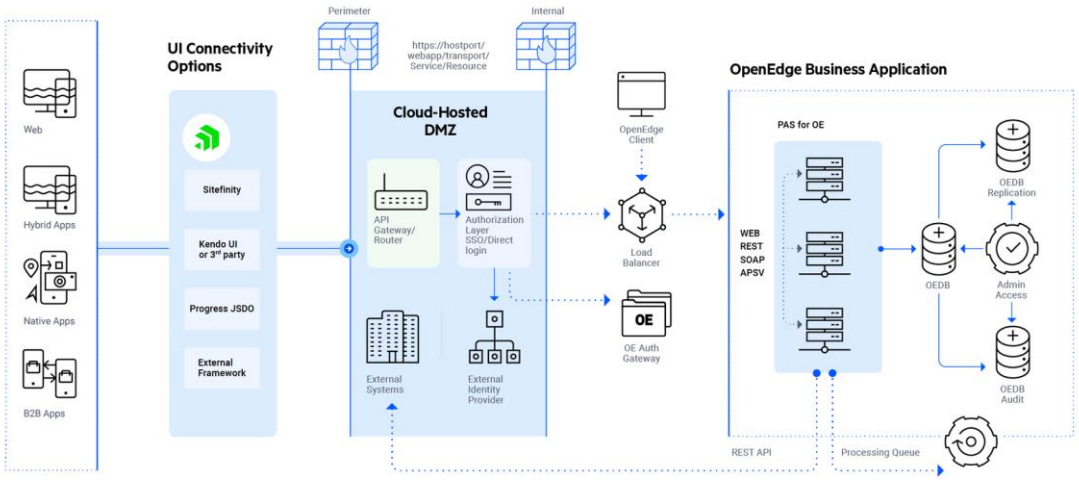
- S'appuyer sur un type de serveur d'application "standard" et reconnu Apache Tomcat dont l'architecture est connue par les clients et partenaires Progress.
- Exécuter des traitements ABL OpenEdge
- Exposer des WebServices SOAP
- Exposer des Services REST
- Avoir une composante Serveur HTTP et permettre de gérer nativement les requêtes et réponses HTTP.
- Implémenter des éléments de sécurité standards de l'industrie

Ce nouveau serveur d'application remplace donc avantageusement le couple JSE + Appserver dans des architectures de Services actuelles.

En 2022 , la version courante de ce serveur d'Application PASOE est 12.5 (février 2022).

En terme d'architecture , PASOE permet d'envisager une architecture évolutive permettant d'avoir une réelle exposition de services métier ou services techniques OpenEdge ABL dans une architecture ouverte

Architecture Ouverte orientée Cloud



Appserver et PASOE : 2 Serveurs différents

L'attente de tous les développeurs et utilisateurs du serveur Appserver était d'avoir une continuité et une pérénnité dans l'architecture.

La force de Progress sur OpenEdge a toujours été de permettre des montées de versions assurant une compatibilité.

Ceci est vrai pour les migration d'application et la compatibilité du code depuis la création du langage Progress (L4G Progress jusqu'à la version 9 puis OO ABL depuis openEdge 10).

Comme exprimé précédemment, l'Appserver Progress est un serveur d'application exécutant uniquement des applications OpenEdge.

De plus l'architecture technique de l'Appserver impose de rajouter des composants externes pour une architecture orientée service.

PASOE a été conçu pour jouer un rôle de serveur d'application "unifié".

Pour ce serveur 4 composantes principales on été créée qui correspondent à la notion de "Transport" :

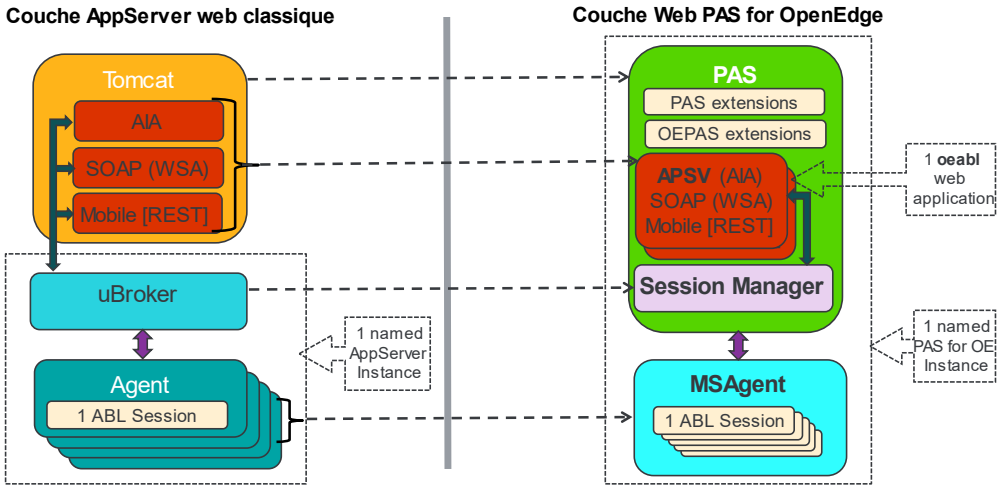
- WEB : Gestion de requêtes HTTP et site Web
- REST : Exposition de Services REST
- SOAP : Exposition de Webservice SOAP
- APSV : Exécution de traitements OpenEdge (correspondant à l'Appserver via HTTP/tunneling)

Le serveur Appserver est utilisable de la version V9 Progress jusqu'à OpenEdge 11.7.

Le serveur PASOE est utilisable à partir de la version 11.5.

A partir de la version OpenEdge 12.0 seul le seveur PASOE est disponible.

PASOE : Architecture multi sessions



Appserver

1 agent correspond à l'exécution d'un process PROAPSV qui exécute 1 session.

En général on démarre un Broker appserver en précisant un nombre d'agents à démarrer.

Il y a un mécanisme de type "Round Robin" mis en oeuvre et dans un appel de type "StateLess", chaque agent est sollicité à tour de rôle.

Si le nombre de requêtes ne nécessite pas d'avoir le nombre initial d'agents démarrés, il y a un arrêt de certains agents.

Si la charge augmente et que les Agents sont saturés, alors des agents supplémentaires sont démarrés (à concurrence du nombre maximum d'agents)

PASOE : Gestion des requêtes

Le mécanisme de gestion des requêtes dans PASOE est différent.

Un agent multi sessions (msagent) MPROAPSV est démarré.

Chaque msagent peut gérer des contextes d'exécution dans des sessions.

Le nombre maximum de sessions par msagent est de 200.

Dans le cas de démarrage de plusieurs msagent, il n'y a de mécanismes de répartition des requêtes sur les différentes msagents.

Toutes les requêtes sont traitées par les sessions du premier msagent jusqu'à saturation avant de solliciter d'autre msagent.

Gestion de Mémoire : Appserver / PASOE

Appserver

Si une requête exécutée par un agent ne libère pas bien les objets mémoires, cela n'aura pas d'impact sur les autres agents.

PASOE

Si une requête exécutée, sur un msagent utilisant une session, ne libère pas les objets, il y potentiellement une augmentation de la mémoire utilisée par le msagent.

Ceci peut se visualiser par une augmentation progressive de l'utilisation mémoire.

Tests de migration Appserver vers PASOE : A Retenir

Dans tous les cas de figure le passage d'un Appserver vers PASOE ne posera pas de problème particulier au niveau fonctionnement.

Par contre, PASOE implique une meilleure gestion de la libération des objets mémoires.

Dans le cas de migration, tests à effectuer

- Tests fonctionnels : pour valider que les traitements s'éxécutent correctement et avec des temps de réponses corrects.
- Tests de montée en charge : pour valider qu'il n'y a pas de problèmes de fuite mémoire

Comparaisons et Avantages PASOE

	Appserver Classique	PASOE
Exécution ABL	Oui	Oui
Exposition SOAP	Non (nécessite JSE *)	Oui
Exposition REST	Non (nécessite JSE *)	Oui
Web Server intégré	Non	Oui
Déploiement WebApp	Non	Oui
Agent MultiThread	Non	Oui
Instances liées à une version OpenEdge	Non	Oui
Gestion de tous les attributs HTTP (verbes , header, …)	Non	Oui (Webhandler)
Utilisation avec Webclient	Oui	Oui
Agent Indépendant du mode d'utilisation	Non	Oui
Gestion obligatoire via l'Admin Service Progress	Oui	Non
Mécanismes Load Balancing	Non	Oui
PASOE sous forme d'image Docker	Non	Oui
Mécanismes de Sécurité standards	Non	Oui

(*) JSE : Java Servlet Engine (exemple Apache Tomcat)

Les cas de migration Appserver vers PASOE

Recommendations

Dans tous les cas de migration Appserver vers PASOE, les tests fonctionnels et de non régressions sont nécessaires mais ne sont pas suffisants du fait de la nouvelle architecture msagent (Multi Sessions Agent).

Il sera important de tester également la montée en charge (voir paragraphe sur les Fuites Mémoires).

Les cas de migration d'une architecture client/serveur

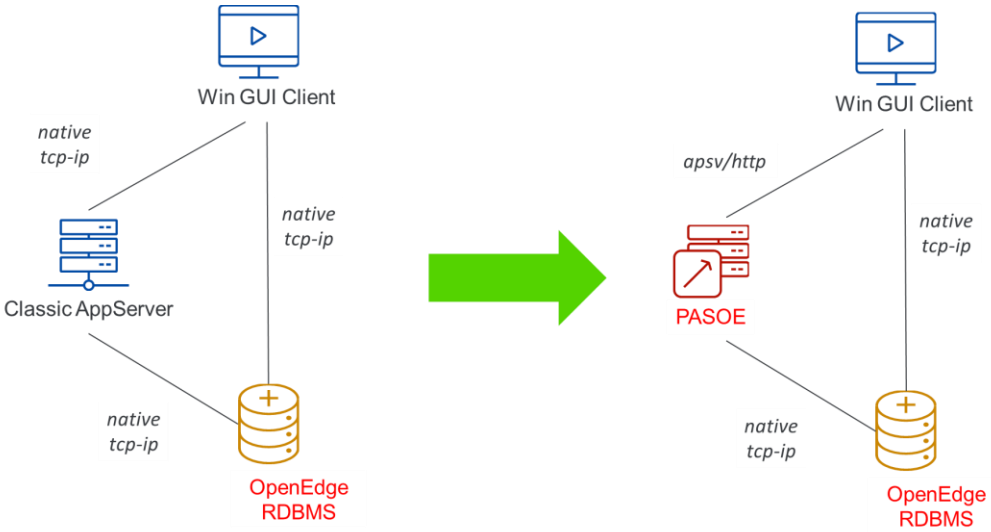


Fig. 1

Dans ce schema, l'application est de type "hybride" avec une partie en client/serveur lourd (Architecture 2 Tier) et une exécution de logique métier sur le serveur d'application.

Le poste client est installé avec la composante Client Networking OpenEdge.

Un accès à la base de données est assuré via le poste client mais aussi via le serveur d'application.

Dans ce cas, les requêtes à la base de données peuvent être réalisées soit à partir du poste client , soit à partir du serveur d'application.

Si le serveur d'application et la base de données se trouvent sur le même serveur alors les requêtes "serveur" peuvent se faire en utilisant la mémoire partagée.

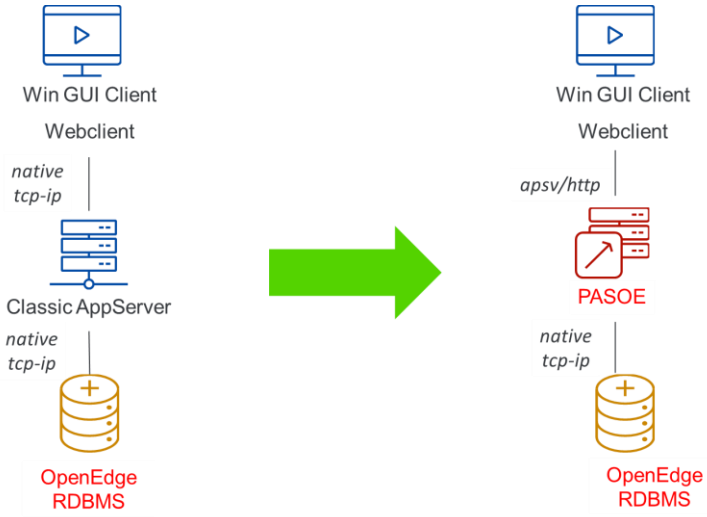


Fig.2

Dans ce schéma , tous les les accès sur la base de données sont assurés via le serveur d'application.

Il n'y a plus de connexion de type client/serveur à partir du poste client.

L'application OpenEdge peut s'exécuter sur le poste Windows en utilisant soit le composant Client Networking OpenEdge soit le composant Webclient OpenEdge .

Modes de connexion au serveur d'application

Appserver	PASOE
Define variable happ as handle. Create server happ. happ:connect("-AppService asbroker1", "" , "")	Define variable happ as handle. Create server happ. happ:connect("-URL http://localhost:8810", "" , "")

Le mode de connexion est le premier élément à modifier dans le cas de la migration. Ceci permettra l'exécution des traitements sur PASOE.

On remarque que la connexion est de type HTTP qui est la transposition de l'architecture mise en oeuvre avec l'Appserver pour exécuter des application OpenEdge Windows communiquant via un protocole HTTP. Ceci s'appuyant sur de l'HTTP Tunneling appelé. Ceci avait donné lieu à la cr"ation d'un servlet appelé AIA (Appserver Internet Adapter)

Entre une communication avec un serveur Appserver et un serveur PASOE pour des applications Windows clientes, il y a quelques éléments à prendre en comptes qui peuvent impacter des temps de réponses

Tableau 1 : Points d'attentions
1. Chaque connexion client APSV est un minimum de 2 requêtes HTTP. 2. Chaque demande de procédure distante client correspond à 1 (ou plusieurs) requêtes HTTP - selon le type de client APSV et s'ils implémentent les améliorations de performances qui ont été apportées dans le cadre de l'AIA. 3. Chaque procédure persistente exécutée par le client APSV ajoute 2 autres messages HTTP pour l'exécution et la suppression de l'objet Procédure persistente du serveur. 4. La déconnexion est un autre message HTTP .
Conséquences possibles : temps de réponse
La surcharge HTTP supplémentaire peut être minime pour une application qui se connecte une fois et n'exécute pas de procédures persistantes. Une application qui se connecte/déconnecte à plusieurs reprises, ou qui exécute/supprime de nombreuses procédures persistantes, peut entraîner une grande quantité de surcharge.

Les cas de migration d'une architecture Web

Appserver - Webspeed

Ceci correspond à des applications Web s'appuyant sur l'approche dites Webspeed qui permettait de développer des procedures L4G pouvant être utilisées directement des applications Web.

L'Appserver permettait d'exécuter ces traitements avec la notion de Speedscript.

Speedscript : permettait d'inclure du traitement ABL dans une page HTML.

Exemple SpeedScript

```
<script language="SpeedScript">
   DEFINE VARIABLE vcCustName AS CHARACTER NO-UNDO.
   DEFINE VARIABLE vcToday AS CHARACTER NO-UNDO.

    FIND FIRST CUSTOMER NO-LOCK NO-ERROR.

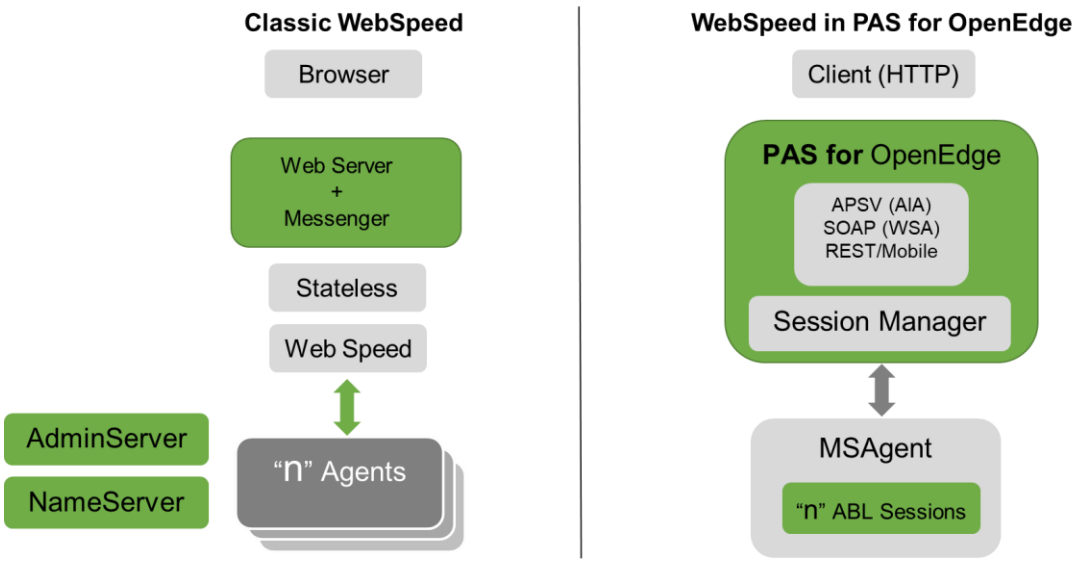
    IF AVAILABLE CUSTOMER THEN DO:
 ASSIGN vcCustName = Customer.Name.
   END.
   ELSE DO:
        ASSIGN vcCustName = "No Customer Available".
   END.

    ASSIGN vcToday = STRING(today).

 </script>
```

Un serveur Web HTTP (IIS , Apache …) était nécessaire pour les pages Web statiques et un composant CGI (Messenger) permettait la communication.

Classic WebSpeed vs. PAS for OpenEdge



PASOE

Avec PASOE il n'y a plus besoin de serveur HTTP.

Un composant de compatibilité (Compatibility Webhandler) permet rapidement de migrer les applications Webspeed vers PASOE.

Avec PASOE, tout type de composantes HTTP peut être gérée via des Webhandler qui vont permettre en ABL de gérer

- Les requêtes HTTP
    - Gestion des verbes HTTP (Get, post…)
- Les réponses HTTP
- Les entêtes HTTP

Exemple kbase Progress pour la migration :
https://knowledgebase.progress.com/articles/Knowledge/how-to-migrate-classic-webspeed-application-to-pas

Conclusions :

La migration à iso fonctionnalités d'une application Web Webspeed est relativement simple mais il est intéressant de revoir les concepts des applications Web modernes s'appuyant sur des Framework Javascripts types Angular, React ou Vue avec une approche s'appuyant sur des API REST

Les cas de migration d'une architecture exposant des Services SOAP

L'utilitaire PROXYGEN est toujours utilisé pour générer les fichiers .wsm et le déploiement des Webservices SOAP.

Créer un Webservice SOAP via Proxygen	https://knowledgebase.progress.com/articles/Knowledge/How-to-create-a-WebService-for-PASOE-in-ProxyGen
Déployer un Webservices via fichier .wsm	https://knowledgebase.progress.com/articles/Knowledge/how-to-deploy-wsm-file-in-pasoe-from-proenv

Les cas de migration d'une architecture exposant des Services REST

Ceci concernerait le cas de services REST déployés via le RestManager (serveur JSE) pour accéder à des traitements s'exécutant sur l'Appserver.

Ceci type de migration pourra être réalisé à iso fonctionnalité mais de nouvelles fonctionnalités comme le WebHandler n'existait pas avec le RestManager.

Cette fonctionnalité est une large amélioration des possibilités de gestion des requêtes/réponses HTTP lors d'appels REST.

Les cas de migration d'une architecture utilisant l'Openclient

Ce cas est à rapprocher avec la migration d'une application Client/serveur utilisant une IHM OpenEdge (Client Networking) car le transport APSV est utilisé.

Dans le cas de l'utilisation Openclient Java ou OpenClient .NET il faudra suivre les préconisations sur les librairies à utiliser sur la partie cliente qui assure les appels PASOE.

De la même manière que pour les applications client/serveur, il sera important de tester le fonctionnel et la montée en charge (voir paragraphe sur les Fuites Mémoires)

PASOE et API : Swagger , JMX , Healthscanner

Swagger

PASOE expose des API d'administration via une interface Swagger.

L'accès à l'interface Swagger : http://servername:8810/oemanager/doc/api-docs?url=/oemanager/doc/openapi.json#/ (servername = adresse du serveur hébergeant l'instance PASOE).

Kbase pour activer l'interface : https://knowledgebase.progress.com/articles/Article/How-to-enable-Swagger-for-PASOE-instance

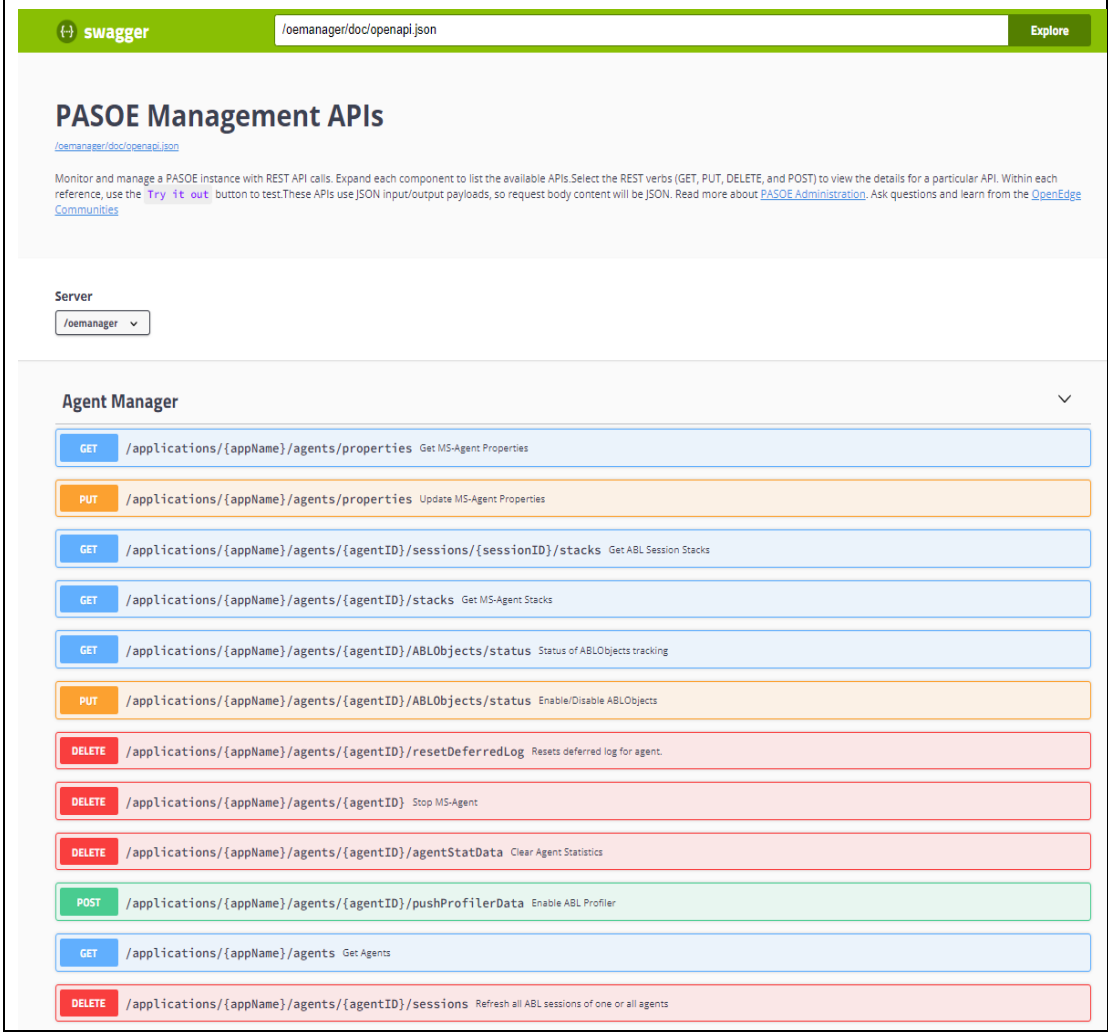
Ceci permet de rapidement récupérer des informations sur le fonctionnement en production.

Une fonctionnalité appelée Healthscanner permet d'avoir des informations sur le bon état de santé d'une configuration PASOE.

L'accès via une console JMX permet également de collecter des informations.

Exemple écrans swagger sur une instance oepas1 localhost

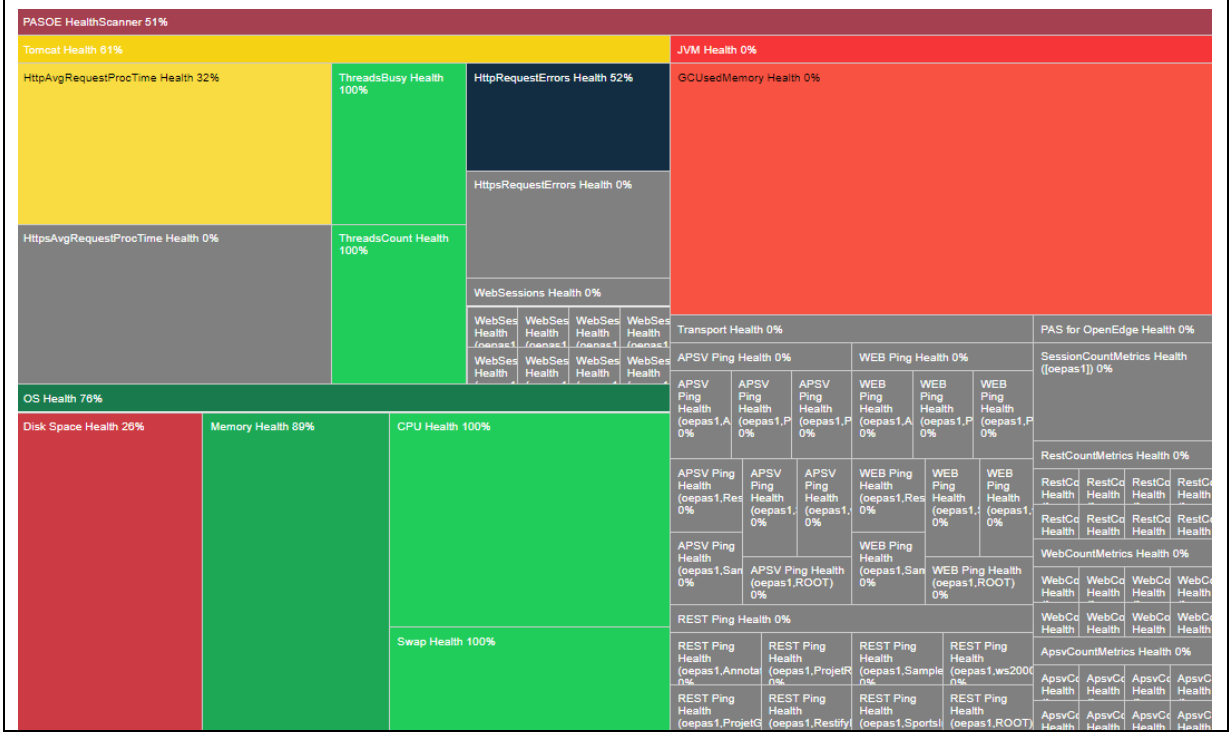
http://localhost:8810/oemanager/doc/api-docs?url=/oemanager/doc/openapi.json#/



Healthscanner : donne une vision instantanée de l'état d'un serveur PASOE

Accessible via une URL de type http://servername:8899/heatmap.html

Exemple Healthscanner



http://localhost:8899/health?view=summary : avec un exemple de PASOE présentant un problème global (état optimal doit être proche de 100%)

{

- **health**: 0.5152988746587488,
- **timestamp**: "2022-05-20T12:58:53.432250100Z",
- **interval**: 60

}

http://localhost:8899/health?view=details : accès aux details des informations

{

- **name**: "PASOE HealthScanner",
- **color**: "#A44050",

- **timestamp**: "2022-05-20T13:00:53.479211300Z",
- **result**:

  {

    - **health**: 0.51565423186049 54,
    - **value**: 0.51565423186049 54,
    - **marginal**: true,
    - **critical**: true,
    - **ignore**: false,
    - **failOnCritical**: false,
    - **failedBy**: null

  },

- **config**:

  {

    - **calculation**: "sum of probes(health * weight) / sum of probes(weight)",
    - **critical**: 0.6,
    - **probeID**: 1,
    - **name**: "PASOE HealthScanner",
    - **marginal**: 0.7,
    - **description**: "Composite weighted health of PAS for OpenEdge server metrics",
    - **weight**: 1,
    - **className**: "com.progress.appserv.services.health.probe.CompositeHealthProbe",
    - **enabled**: true,
    - **failOnCritical**: false

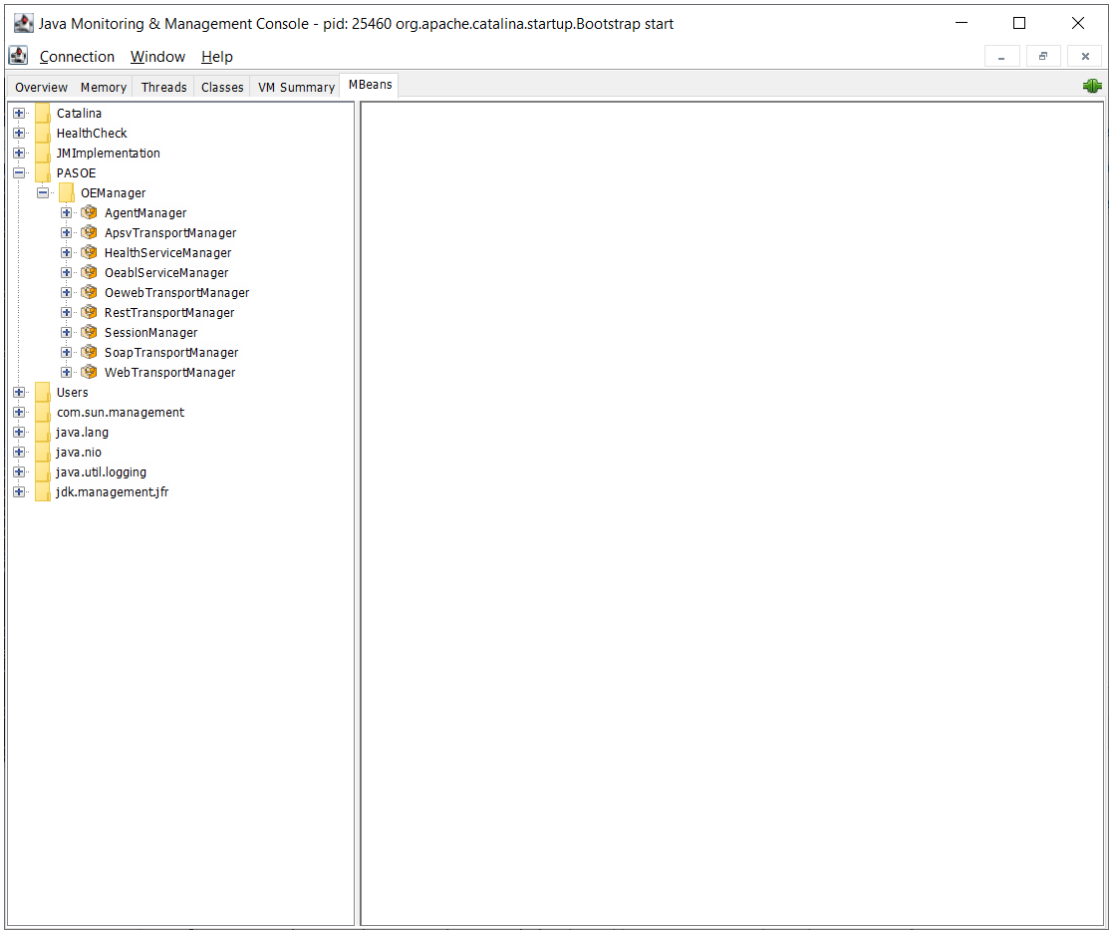
  },

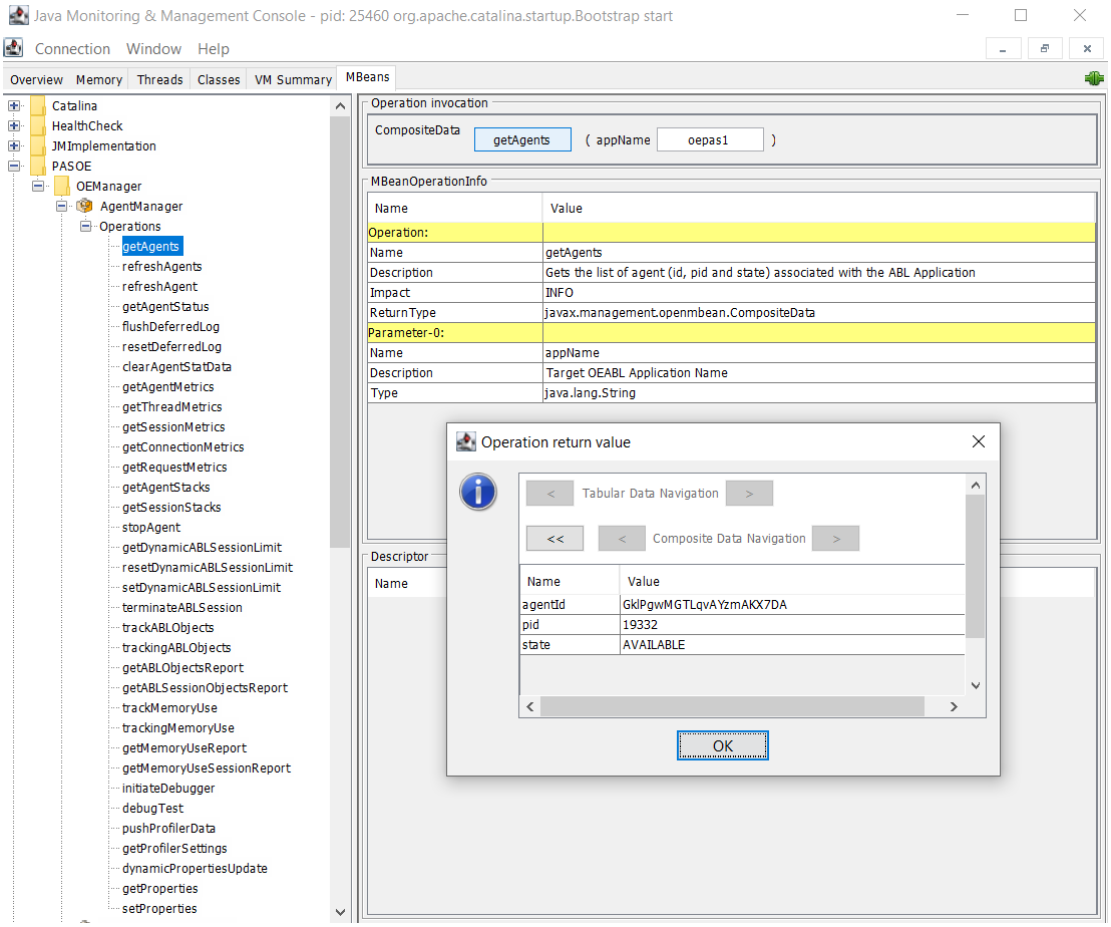
- **probes**:

```
[
    {
        name: "JVM Health",
        color: "#F63538",
        result:
        {
            health: 0,
            value: 0,
            marginal: true,
            critical: true,
            ignore: false,
            failOnCritical: false,
            failedBy: null
        },
        Etc…
```

JCONSOLE

Des informations de fonctionnement sont disponibles via jconsole


Java Monitoring & Management Console - pid: 25460 org.apache.catalina.startup.Bootstrap start
Connection Window Help
Overview Memory Threads Classes VM Summary MBeans
Catalina
HealthCheck
JMImplementation
PASOE
OEManager
AgentManager
ApsvTransportManager
HealthServiceManager
OeablServiceManager
OewebTransportManager
RestTransportManager
SessionManager
SoapTransportManager
WebTransportManager
Users
com.sun.management
java.lang
java.nio
java.util.logging
jdk.management.jfr



PASOE vs Appserver : Performances

Comme explicité plus avant dans ce document , lors d'une migration appserver vers PASOE, les tests fonctionnels incluant des tests de non régressions sont nécessaires mais pas suffisants.

Applications client/server ou openclient / transport APSV

Certains facteurs peuvent jouer sur les performances et de façon plus sensible dans le cas d'applications utilisant le transport APSV.

Ce transport est directement lié à des applications client/serveur ou une interface JAVA ou .NET (openclient) utilisant PASOE dans une composante N-TIER (traitements métier ABL).

Dans le cadre de projets , des tests on été effectués présentant une certaine stabilité voir légère amélioration dans les performances dans le cas de migration Appserver vers PASOE.

Comme l'Appserver n'existe plus à partir de OpenEdge 12, seuls des comparatifs OpenEdge Appserver 11.7 et PASOE 12.x sont possibles.

Certains apports complémentaires liés à la base de données OpenEdge 12 et à des aspects liés aux systèmes d'exploitation interviennent également.

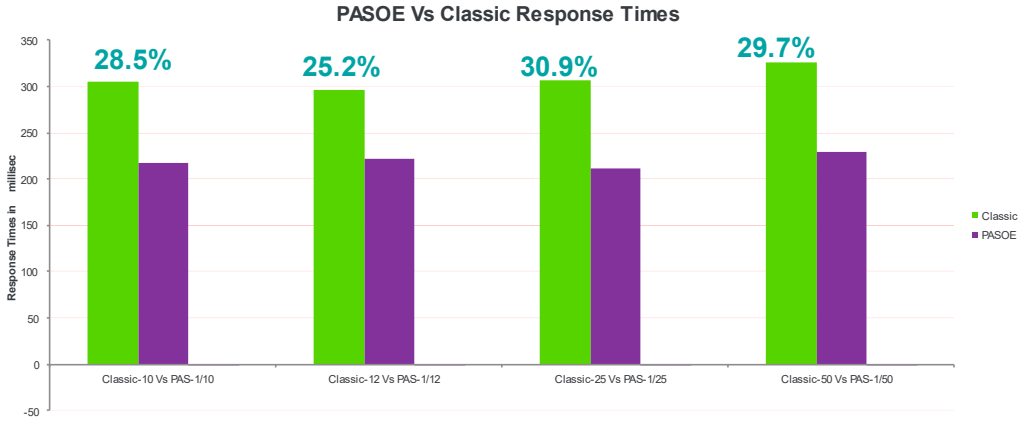
Conclusions:

Donc dans le cas d'utilisation du transport APSV , une phase de tests est largement conseillée pour valider les performances

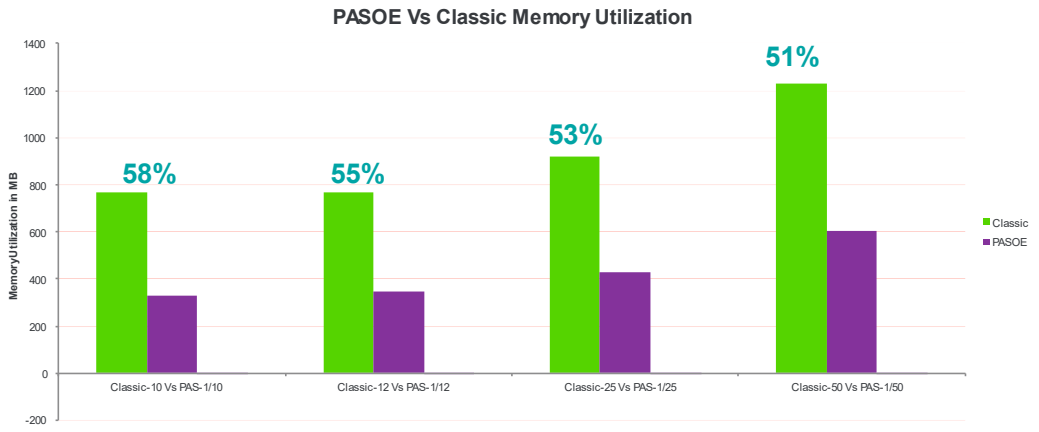
Applications Web / transport REST

Des tests effectués ont montré des gains de performances notables liés à l'architecture PASOE

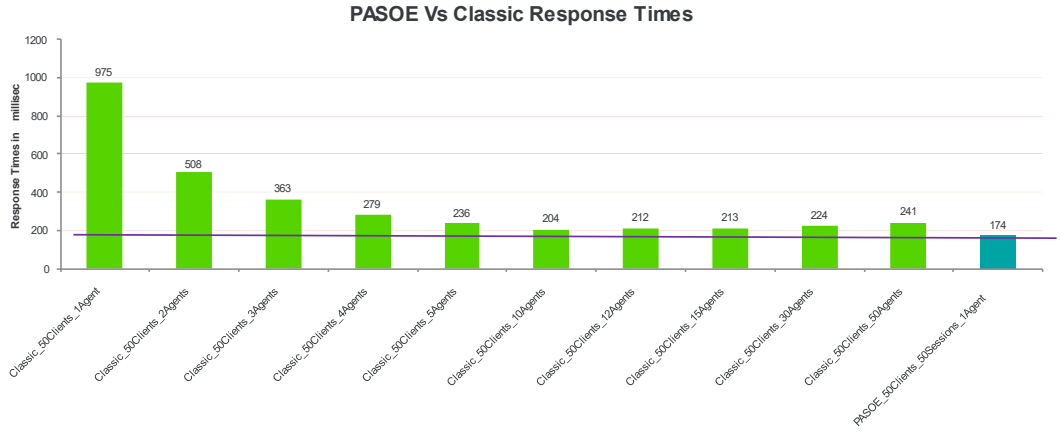
Progress Application Server for OpenEdge - Performance - 25KB of Payload



Progress Application Server for OpenEdge - Performance - 25KB of Payload on Linux



Progress Application Server for OpenEdge - Performance - 5.3KB of Payload on Linux



Progress Application Server for OpenEdge - Performance

	Classic AppServer	PAS for OE	Difference
Scalability			
Client connections	221	1312	493%
Server Resources			
CPU	10 CPUs	5.2 CPUs	192%
Memory	2.1 GB	670 MB	313%
Transactions	203 tps	1698 tps	736%
Client performance			
OpenEdge	472ms	340ms	138%

Conclusions :

PASOE est le transport REST est une combinaison qui apporte le meilleur gain en performances

Appserver Classique et PASOE : Points à considérer

Les appels de traitements sur un Appserver Classique dans le cadre d'une architecture N-TIER (IHM OpenEdge ou utilisation Openclient) sont les bases de comparaisons pour les performances souvent utilisés.

Il faut prendre en compte que les connexions sur PASOE se font via HTTP.Dans le cadre d'applications N-TIER (IHM OpenEdge ou openclient) c'est PASOE avec le transport APSV qui est à rapprocher avec une architecture Appserver utilisant l'AIA (Appserver Internet Adapter).

Dans certains cas de figures, les performances PASOE APSV peuvent ne pas être identiques aux performances Appserver Classique (11.7) avec un accès natif.

Certains tests ont montré que les appels REST OpenEdge 12 sont plus performants que les appels N-TIER avec l'appserver Classique et plus performants que les appels N-TIER (APSV) en OpenEdge 12.

Comme pour tout test de performance, il faudra effectuer des mesures sur l'application à migrer.

De plus , les apports liés à la base de données Enterprise en OpenEdge 12 vont apporter des gains de performances par rapports aux bases de données de version <= 11.7.

PASOE : Gestion de Fuites Mémoire

La gestion de la mémoire est un point de surveillance sur l'architecture multi Sessions du serveur PASOE.

Des fonctionnalités existent afin de détecter ce type de comportement et en particulier sur des objets qui sont créés dynamiquement.

Un cas typique de fuite mémoire peut être repéré si la mémoire utilisée par un processus msagent (Agent Multi Session) ne cesse de croître.

En situation normale, la mémoire d'un msagent doit augmenter pour atteindre un seuil.

Dans le cadre d'une migration Appserver vers PASOE des mécanismes OS permettent d'arrêter des process si un seuil est atteint.

Linux Out Of Memory killer assure ce fonctionnement et si l'on observe ce comportement sur PASOE, il y a de fortes probabilités d'une fuite mémoire.

En cas de problème de ce type, les actions à mettre en place :

- à court terme "Palliatif" : monitoring des instances PASOE pour mettre en place une solution qui permette de libérer "proprement" (gracefully stop) un msagent. Ceci permet de terminer les requêtes en cours et les nouvelles requêtes sont adressées à un second agent.
  Il faut pour cela que le minimum de msagent soit de 2.
  Au démarrage de l'instance PASOE en Production , 2 msagents sont démarrés.
  Dans le cas d'un problème sur le premier msagent, l'arrêt "Graceful" est initié, et les nouvelles requêtes sont dirigées sur le second msagent.
  Quand le premier msagent se termine, un nouvel msagent est automatiquement démarré pour assurer le minimum de nombre d'agents de 2.
  Un nouveau cycle est engagé.

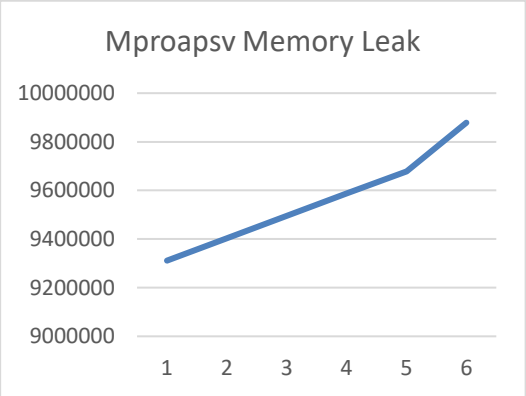
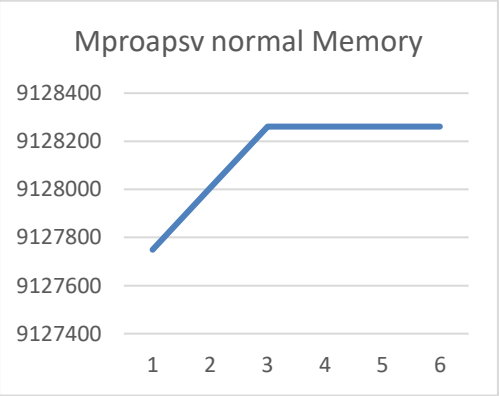
  Une entrée de la kbase Progress fournie un exemple de solution :
  https://knowledgebase.progress.com/articles/Knowledge/Monitoring-PASOE-REST-application-best-practices

L'utilisation des APIs REST de l'oemanager permettent de créer ses scripts ou d'utiliser l'interface Swagger.
https://knowledgebase.progress.com/articles/Knowledge/How-to-trim-and-restart-PASOE-Agents-using-REST-APIs

- À moyen terme "Correctif" : mettre en place des outils qui permettent de monitorer les utilisations mémoires pour détecter les fuites mémoires y compris sur les objets dynamiques.

Dans tous les cas, l'ouverture d'un Ticket au support technique sera indispensable car les cas référencés permettent d'accélérer la resolution.

Exemple de traitement avec Fuite mémoire et sans Fuite Mémoire

Analyse consommation msagent avec fuite mémoire (graphique 1)	Analyse consommation msagent sans fuite mémoire (graphique 2)
	

Exemple de code provoquant une fuite mémoire (graphique 1)

```
DEFINE VARIABLE tth4 AS HANDLE.
DEFINE VARIABLE btth4 AS HANDLE.
DEFINE VARIABLE qh4 AS HANDLE.
DEFINE VARIABLE bCust AS HANDLE.
DEFINE VARIABLE bOrder AS HANDLE.
DEFINE VARIABLE i AS INTEGER.
DEFINE VARIABLE fldh AS HANDLE EXTENT 15.

bCust = BUFFER customer:HANDLE.
bOrder = BUFFER order:HANDLE.
```

```
CREATE TEMP-TABLE tth4.
tth4:ADD-FIELDS-FROM(bCust,"address,address2,phone,city,comments").
tth4:ADD-FIELDS-FROM(bOrder,"cust-num,carrier,instructions,PO,terms").
tth4:TEMP-TABLE-PREPARE("CustOrdJoinTT").
btth4 = tth4:DEFAULT-BUFFER-HANDLE.

FOR EACH customer WHERE cust.custnum < 1000, EACH order OF customer:
btth4:BUFFER-CREATE.
btth4:BUFFER-COPY(bCust).
btth4:BUFFER-COPY(bOrder).
END.

/* Create Query */
CREATE QUERY qh4.
qh4:SET-BUFFERS(btth4).
qh4:QUERY-PREPARE("for each CustOrdJoinTT").
qh4:QUERY-OPEN.

REPEAT WITH FRAME zz DOWN:
qh4:GET-NEXT.
IF qh4:QUERY-OFF-END THEN LEAVE.
REPEAT i = 1 TO 15:
fldh[i] = btth4:BUFFER-FIELD(i).
/*DISPLAY fldh[i]:NAME FORMAT "x(15)"
fldh[i]:BUFFER-VALUE FORMAT "x(20)".*/
END.
END.

btth4:BUFFER-RELEASE.
//DELETE OBJECT tth4.

//DELETE OBJECT qh4.
```

Sur cet exemple les objets tth4 et qh4 ne sont pas détruits et les appels successifs consomment de la mémoire supplémentaire

Exemple de code correct (graphique 2)

```
DEFINE VARIABLE tth4 AS HANDLE.
DEFINE VARIABLE btth4 AS HANDLE.
DEFINE VARIABLE qh4 AS HANDLE.
DEFINE VARIABLE bCust AS HANDLE.
DEFINE VARIABLE bOrder AS HANDLE.
DEFINE VARIABLE i AS INTEGER.
DEFINE VARIABLE fldh AS HANDLE EXTENT 15.

bCust = BUFFER customer:HANDLE.
bOrder = BUFFER order:HANDLE.

CREATE TEMP-TABLE tth4.
tth4:ADD-FIELDS-FROM(bCust,"address,address2,phone,city,comments").
```

```
tth4:ADD-FIELDS-FROM(bOrder,"cust-num,carrier,instructions,PO,terms").
tth4:TEMP-TABLE-PREPARE("CustOrdJoinTT").
btth4 = tth4:DEFAULT-BUFFER-HANDLE.

FOR EACH customer WHERE cust.custnum < 1000, EACH order OF customer:
btth4:BUFFER-CREATE.
btth4:BUFFER-COPY(bCust).
btth4:BUFFER-COPY(bOrder).
END.

/* Create Query */
CREATE QUERY qh4.
qh4:SET-BUFFERS(btth4).
qh4:QUERY-PREPARE("for each CustOrdJoinTT").
qh4:QUERY-OPEN.

REPEAT WITH FRAME zz DOWN:
qh4:GET-NEXT.
IF qh4:QUERY-OFF-END THEN LEAVE.
REPEAT i = 1 TO 15:
fldh[i] = btth4:BUFFER-FIELD(i).
/*DISPLAY fldh[i]:NAME FORMAT "x(15)"
fldh[i]:BUFFER-VALUE FORMAT "x(20)".*/
END.
END.

btth4:BUFFER-RELEASE.
DELETE OBJECT tth4.
DELETE OBJECT qh4.
```

Le fait de préciser la suppression des objets (delete object tth4 et qh4) après utilisation libère la mémoire

La libération des objets en mémoire peut être pilotée via des procédures associées au serveur d'application (procedures Deactivate , Disconnect ou Shutdown)

Activate	**Deactivate**
Connect	**Disconnect**
Startup	**Shutdown**

PASOE et Répartition de Charges

Une solution de répartition de charges peut être utile pour :

- Répartir les requêtes vers plusieurs instances PASOE
- Pallier une défaillance d'une instance PASOE et garantir la disponibilité d'un service
- Mettre en oeuvre une solution de montée de version dans une architecture de haute disponibilité (24x7)

L'Appserver Classique peut disposer d'une répartition de charge en ajoutant un composant appelé Name Server Load Balancer.

Pour PASOE il n'y a pas de correspondence du Name Server Load Balancer (en d'autres termes le Name Server Load Balancer ne peut être utilisé avec PASOE).

Des configurations peuvent être mises en place en utlisant des fonctionnalités liés à PASOE.

Divers solutions peuvent être mises en place :

- Serveur HTTP frontal
- Solution de type F5
- Solutions Citrix
- Progress Kemp Loadmaster
- …

PASOE et LOADMASTER

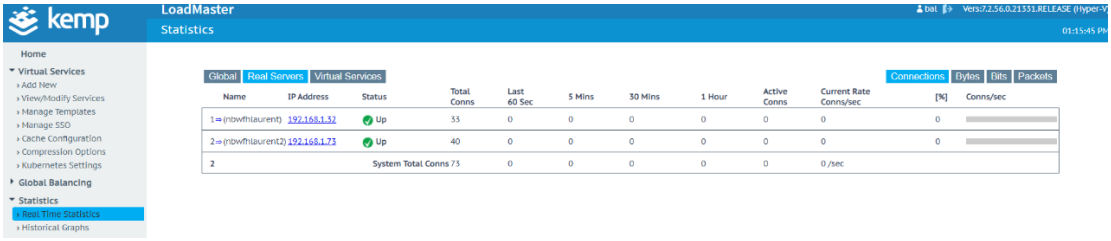
Loadmaster est une solution de répartition de charge et de sécurité qui provient du rachat de la société Kemp par Progress Software en 2021.

La solution intervient sur 3 aspects

Loadmaster peut être disponible sous format hardware ou alors logiciel au travers de machines virtuelles.

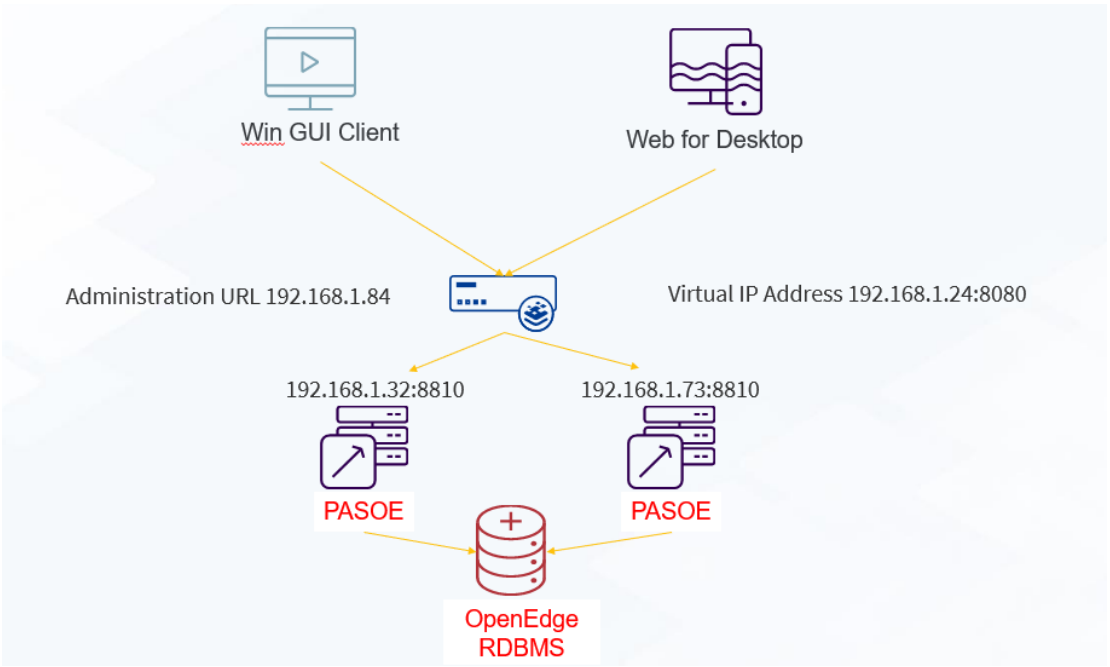
Dans le cadre de PASOE la configuration intuitive de Loadmaster permet rapidement de mettre en oeuvre la répartition de charge que ce soit pour l'utilisation dans une architecture N-TIER (transport APSV) ou alors l'exposition de Services REST (transport REST ou WEB).

Une console d'administration Web permet de paramétrer ce qui est necessaire.



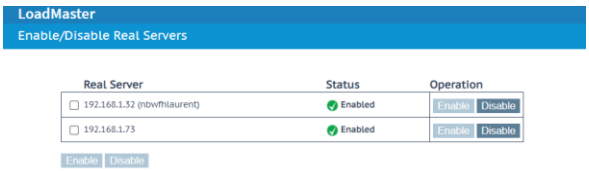
Le composant Loadmaster mettra en oeuvre une IP Virtuelle ainsi qu'un port d'accès

Les instances PASOE peuvent se dérouler avec des ports différents et l'architecture pourra ressembler au schéma ci-dessous



Hormis l'aspect répartition de charge, la console d'administration permet de désactiver un noeud (sans arrêter le serveur PASOE) afin de procéder à des opérations de maintenance :

- Mise à jour de version OpenEdge
- Mise à jour d'une version d'application
- Mise à jour système
- …



Dans le cadre d'une application utilisant le transport APSV, il faudra modifier la chaine de connexion pour accéder à l'IP virtuelle et le port définis pour le Virtual Service Loadmaster.

Ceci sera applicable pour les applications OpenEdge N-TIER y compris les application utilisant Openclient Java ou .NET

Exemple :

```
DEFINE DATASET dsOrder FOR ttOrder, ttOLine, ttItem

    DATA-RELATION drOrderLine FOR ttOrder, ttOLine RELATION-FIELDS
(OrderNum,OrderNum)

    DATA-RELATION drLineItem FOR ttOLine, ttItem RELATION-FIELDS
(ItemNum,ItemNum).


CREATE SERVER happ.

retok = happ:CONNECT("-URL http://192.168.1.24:8080/apsv" ,"" ,"").

MESSAGE retok

    VIEW-AS ALERT-BOX INFORMATION BUTTONS OK.

RUN BenchAPSV_1.p ON happ (INPUT 1 , OUTPUT DATASET DSORDER).

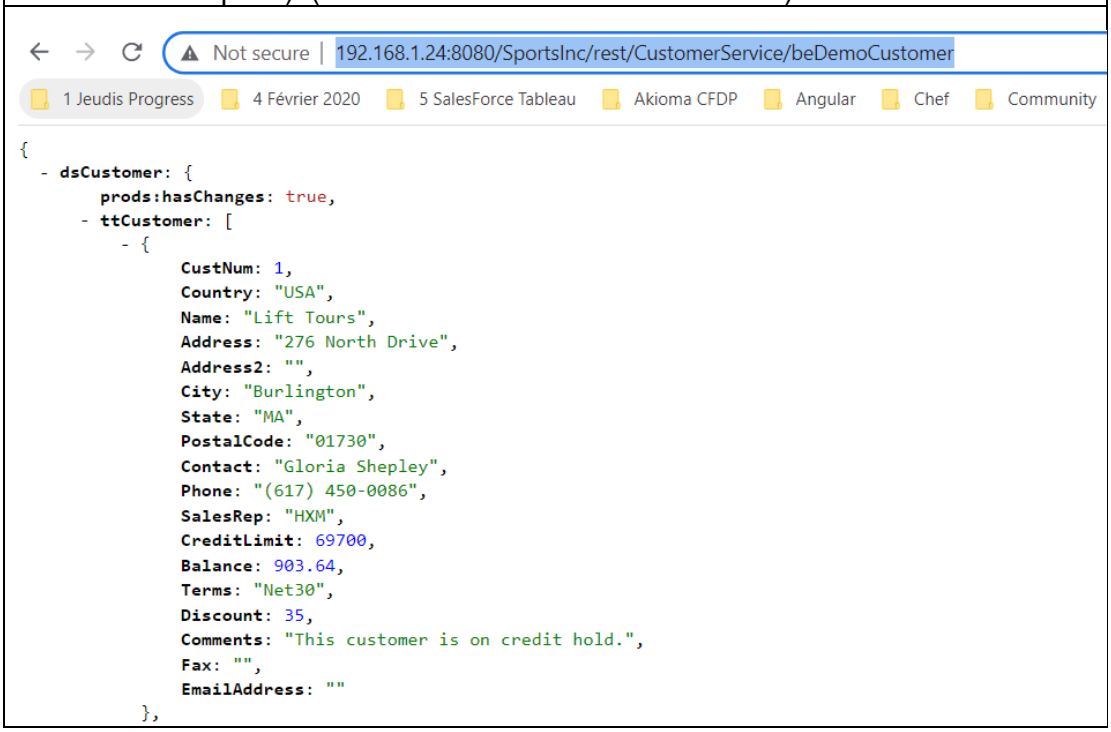
FIND FIRST ttorder.

DISP ttorder.
```

Loadmaster pourra aussi simplifier des aspects d'authentification dans le cadre d'une stratégie SSO (Single Sign On).

Dans le cas d'un accès à des services REST , il suffira de modifier l'URL afin de pointer sur la combinaison Adresse-IP:Port du service Loadmaster.

Exemple: 192.168.1.24:8080 au lieu d'une address de type 192.68.1.32:8810 (port d'une server oepas1) (192.168.1.32 étant l'IP d'un serveur)



D'autres informations sont disponibles sur
https://support.kemptechnologies.com/hc/en-us/articles/4731674180877-Progress-Application-Server-for-OpenEdge

PASOE et Container Docker

Le serveur d'application PASOE est disponible sous forme d'image docker Linux soit sur le site de téléchargement Progress soit sur DockerHub.

Ceci permet une installation rapide en complétant par un fichier de configuration (Progress.cfg) à fournir pour l'utilisation.

L'utilisation du concept de container sera possible :

- En développement dans le cadre d'une chaine CI/CD et assurer le déploiement en tests

- En production pour rapidement monter un environnement de façon d'exécution pour une version déterminée.

En prélable de l'utilisation de PASOE sous forme de container Docker, il sera important de maîtriser les concepts:

- Docker avec les commandes usuelles. Le maramétrage du fichier de configuration Dockerfile
- Docker-compose : qui permet de configurer dans un fichier de type YAML les composants à utiliser (par exemple : container web + container PASOE..)

Sur le serveur Linux cible il faudra avoir installé Docker et docker-compose. Divers articles parlent de ces installations et les tableaux ci-dessous sont donnés à titre indicatif.

Exemple installation Docker, docker-compose et utilisation image PASOE

Installer docker (exemple Centos) : https://docs.docker.com/engine/install/centos/

Set up the repository

Install the `yum-utils` package (which provides the `yum-config-manager` utility) and set up the repository.

```
$ sudo yum install -y yum-utils


$ sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
```

Install Docker Engine

1. Install the *latest version* of Docker Engine, containerd, and Docker Compose or go to the next step to install a specific version:
2. `$ sudo yum install docker-ce docker-ce-cli containerd.io docker-compose-plugin`

   If prompted to accept the GPG key, verify that the fingerprint matches `060A 61C5 1B55 8A7F 742B 77AA C52F EB6B 621E 9F35`, and if so, accept it.

This command installs Docker, but it doesn't start Docker. It also creates a `docker` group, however, it doesn't add any users to the group by default.

3. To install a *specific version* of Docker Engine, list the available versions in the repo, then select and install:

   a. List and sort the versions available in your repo. This example sorts results by version number, highest to lowest, and is truncated:

   ```
   $ yum list docker-ce --showduplicates | sort -r
   ```

   ```
   docker-ce.x86_64  3:18.09.1-3.el7                     docker-ce-
   stable
   docker-ce.x86_64  3:18.09.0-3.el7                     docker-ce-
   stable
   docker-ce.x86_64  18.06.1.ce-3.el7                    docker-ce-
   stable
   docker-ce.x86_64  18.06.0.ce-3.el7                    docker-ce-
   stable
   ```

   The list returned depends on which repositories are enabled, and is specific to your version of CentOS (indicated by the `.el7` suffix in this example).

   b. Install a specific version by its fully qualified package name, which is the package name (`docker-ce`) plus the version string (2nd column) starting at the first colon (`:`), up to the first hyphen, separated by a hyphen (`-`). For example, `docker-ce-18.09.1`.

   ```
   $ sudo yum install docker-ce-<VERSION_STRING> docker-ce-cli-
   <VERSION_STRING> containerd.io docker-compose-plugin
   ```

   This command installs Docker, but it doesn't start Docker. It also creates a `docker` group, however, it doesn't add any users to the group by default.

4. Start Docker.

5. ```
   $ sudo systemctl start docker
   ```

6. Verify that Docker Engine is installed correctly by running the `hello-world` image.

   ```
   $ sudo docker run hello-world
   ```
   ```

installer docker compose sur centos
https://yallalabs.com/devops/how-to-install-docker-compose-centos-8/

Différents modes de déploiements sont possibles et des exemples sont disponibles sur la base de connaissances Progress ou sur le site communities.

Exemple qui s'appuie sur une entrée de community qui fournit les sources nécessaires. Cet exemple met également en oeuvre Elasticsearch et la console Kibana https://community.progress.com/s/question/0D54Q00007pHA0KSAW/use-the-container-image-for-pas-for-openedge-122x-or-125-with-a-sample-application
Use the Container Image for PAS for OpenEdge 12.2.x or 12.5 with a Sample Application Use the Container Image for PAS for OpenEdge 12.2.x or 12.5 with a Sample Application **Requirements:** • Docker environment • Docker Compose • A valid progress.cfg file • OpenEdge 12.2.x or 12.5 Environment • Scripts to deploy Progress Application Server for OpenEdge Container. (Attached) Note: *For the deployment, we are using service port 8811(sample-app), 9200(elasticsearch), 5601(kibana) and 8080(web-ui). These ports should be available.* See FAQ wiki for additional info: https://community.progress.com/s/question/0D54Q00007pHAYgSAO/docker-container-for-pasoe-faq **Use the Container Image** 1. Start a database server if you don't have a running database server. Below is an example • Create a copy of the sports2000 database and start the database server (broker) ○ prodb sports sports2000

- proserve sports -S <DB_PORT>

2. Build the Sports sample app:

- cd Sports
- For connecting to the database, update the ./conf/startup.pf file with the below content and substituting the required field.
  - -db sports -H <IP_ADDRESS_OF_DB_HOST> -S <DB_PORT>
- In an openedge environment, run the below command
  - ant package
  - Note: A Sports.zip file is generated in ./output/package-output
- Change the working directory to the parent to follow further steps
  - cd ..

3. Start the Elasticsearch, Kibana and web-ui service

- Update the value for serviceURI in webui/grid.js to point to your Docker host.
- Increase virtual memory settings to run Elasticsearch:
  - sudo sysctl -w vm.max_map_count=262144
  - For additional info: https://www.elastic.co/guide/en/elasticsearch/reference/current/vm-max-map-count.html
- Start the services
  - docker-compose up -d
- Check the status of Elasticsearch via a web browser. Elasticsearch may take some time to start.
  - http://<docker-host-machine>:9200
- Check the Elasticsearch starts:
  - docker-compose logs -f elasticsearch

4. Deploy the sample app in PASOE

- Download the PASOE image zip from Electronic Software Distribution (ESD) and unzip it to a folder say pasoe-sample-app. For more details, refer https://docs.progress.com/bundle/pas-for-openedge-docker/
- Change the directory
  - cd pasoe-sample-app
- Copy the Sports.zip generated from Step 2 to ./deploy/ablapps
  - cp ../Sports/output/package-output/Sports.zip ./deploy/ablapps/
- Copy the config.properties from sample app to ./deploy/config.properties
  - cp ../config.properties ./deploy/config.properties
- Update the value for HOST in ../fluentbit/conf/fluent-bit-output.conf to point to your Elasticsearch host.
- Copy the Fluent Bit config from sample app to push the logs to Elasticsearch
  - cp ../fluentbit/conf/fluent-bit-output.conf ./deploy/conf/logging/
- Copy the license file progress.cfg to ./deploy/license
- If you want to change the database during deployment, you can do this by updating the ./deploy/conf/runtime.properties with below content and substituting the requied field. If localhost is used in Step 2 for connecting to the database, please do update the ./deploy/conf/runtime.properties to point to the IP_ADDRESS_OF_DB_HOST.

- Sports.DB.CONNECTION.PARAMS=-db sports -H <IP_ADDRESS_OF_DB_HOST> -S <DB_PORT>
- Deploy the sample app
  - ant -f ./deploy/build.xml deploy

5. Access the PAS for OpenEdge instance via a web browser:

- https://<docker-host-machine>:8811/
- https://<docker-host-machine>:8811/Sports/
- https://<docker-host-machine>:8811/Sports/static/SportsService.json
- https://<docker-host-machine>:8811/Sports/rest/SportsService/Customer
- Note: By default, the PAS for OpenEdge instance will use HTTPS with a test certificate. You will need to accept access with this certificate.

6. Access the web-ui service via a web browser:

- http://<docker-host-machine>:8080

7. Access Elasticsearch to check on available logs

- http://<docker-host-machine>:9200/_cat/indices

8. Access Kibana

- http://<docker-host-machine>:5601
- Notes:
  - Select Management/Index Management to see the indices in Elasticsearch.A index named as pasoe-container-logs should be present.
  - Select Management/Index Pattern to create an index for Kibana:
    - Create index patterns as 'pasoe_container*'
    - Specify @timestamp to filter data by time
  - Select Discover to see pasoe logs. You can search logs for logtype:
    - pasoe_agent_log, pasoe_application_log, pasoe_localhost_log, pasoe_localhost_access_log, start_server_log and etc.

9. Stop the running services

- Stop Elasticsearch, Kibana, and web-ui
  - docker-compose -f ../docker-compose.yaml down
- Stop deployed sample app
  - ant -f ./deploy/build.xml undeploy

NOTE: Update OE version in the build.xml and config.properties file in the deployScripts.zip (12.2.0, 12.2.1, 12.3.0 etc.)

PASOE et Sécurité

Le serveur PASOE est un serveur apache tomcat adapté pour l'utilisation OpenEdge.

Il permet de mettre en oeuvre des éléments de sécurité d'accès qui ne sont pas inclus dans le serveur classique.

Entre autre il facilite l'utilisation LDAP ou l'utilisation de oAuth2 pour la délégation d'authorisation et la mise en place d'une stratégie SSO (single Sign On).

De ce fait il inclut les mécanismes liés au Spring Security Framework.

Dans le cadre de mises à jour de version, il sera important de mettre à jour les instances PASOE car des changements liés à spring Security peuvent être intégrés.

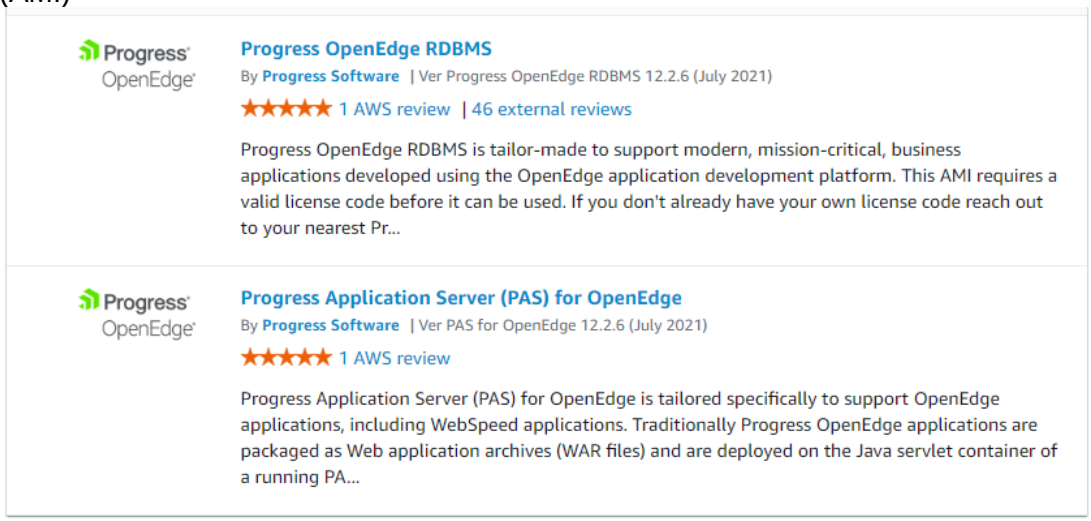
Par exemple entre openEdge 11.7 et 12.x comme expliqué dans la kbase https://knowledgebase.progress.com/articles/Article/PASOE-What-are-the-changes-in-oeablSecurity-properties-between-OE-11-7-and-OE-12-x?popup=true

Extrait

Property	Release	Summary of change	Reason for the change
`OEClientPrincipalFilter.enablecp`	11.7+	Replaced by: `OEClientPrincipalFilter.enabled`	Changed from enablecp to enabled.
`OEClientPrincipalFilter.key`	11.7 Both key and registry file 12.2 Only registry, key was deleted.	**11.7** Added option to use `OEClientPrincipalFilter.registryFile` which override key. **12.2** Removed `OEClientPrincipalFilter.key` option.	Improved security. Registry is an encrypted keystore.
`OERealm.AuthProvider.createCPAuthn`	11.7+	Deleted	Obsolete. No use case for changing.
`OERealm.AuthProvider.sealClientPrincipal`	11.7+	Deleted	Obsolete. No use case for changing.
`OERealm.AuthProvider.key`	11.7+	Deleted	Obsolete. OERealm uses the OEClientPrincipalFilter.registryfile.

PASOE et le Cloud

La migration vers des solutions cloud AWS est facilitée sur les composants OpenEdge et en particulier sur PASOE.
Le Marketplace AWS propose 2 composants OpenEdge Amazon Machine Image (AMI)



Un guide de démarrage et de prise en main “quickstart” est disponible à l’adresse https://aws-quickstart.github.io/quickstart-progress-openedge/ ou https://aws.amazon.com/fr/quickstart/architecture/progress-openedge/

PASOE en Conclusion

PASOE est un serveur d’application obligatoire dans le cadre de la migration vers une version OpenEdge >= 12.

Il n’est définitivement pas une évolution de l’Appserver Classique

Les modes d’utilisation et bonnes pratiques mis en oeuvre pour l’Appserver Classique depuis des années (voire des décennies) sont à revoir et adapter du fait de l’architecture de PASOE.

La mise en oeuvre ne doit pas être sous estimée mais apporte un ensemble de standards qui permettent à des experts de ce type de technologies de rapidement maîtriser ce nouveau composant.

De nouvelles fonctionnalités sont accessibles et la migration d’applications vers le Cloud est facilité.

PASOE contribue à la stratégie Progress de tendre vers une haute disponibilité de 99,999%.