rfc3928

LCUP : Client Update Protocol

Ce protocole permet à un client LDAP de synchroniser le contenu d'un DIT et d'être notifié des changements du contenu. Il adresse les problèmes suivants :

Lorsqu'un client utilise une copie local en lecture seul de l'annuaire et qu'il se connecte au réseau, il synchronise son contenu avec l'annuaire et peut optionnellement recevoir des notifications sur les changements qui se produisent pendant qu'il est en ligne. Par exemple, un client mail peut maintenir une copie local de l'adresse book qu'il synchronise avec la copie master quand le client est connecté au réseau de l'entreprise.

Les applications qui synchronisent les données, une application meta-annuaire par exemple, va récupérer périodiquement une liste de modifications de l'annuaire, construire les changements, et les appliquer aux données stockées.

Les clients qui ont besoin de prendre certaines actions quand une entrée d'annuaire est modifiée. Par exemple, un système de messagerie peut vouloir faire un 'create mailbox' quand une nouvelle personne est créée, et un 'delete mailbox' quand l'entrée est supprimée.

Le problème suivant n'est pas adressé : La synchronisation entre serveurs d'annuaire.

LCUP est conçus de telle sorte que le serveur n'a pas besoin de maintenir l'état des informations spécifique à chaque client. Le serveur peut maintenir des informations d'état additionnels sur la modification des attributs et autres. les clients sont responsable du stockage des informations sur la manière de mettre à jours leur données avec le contenu de l'annuaire. Le client décide quand et où récupérer les changements. LCUP nécessite que les clients initient la session update et "pull" les changements depuis le serveur.

applicabilité

LCUP fonctionne mieux quand les conditions suivantes sont rencontrées :

1) le serveur stocke un certain degré d'informations d'état historique pour réduire la quantité de trafic requis pour les synchronisations incrémentales.

2) le client ne peut pas assumer comprendre le modèle d'information physique (attributs virtuels, opérationnels, subentries, etc.) implémentés par le serveur.

Spécification des éléments du protocole

Universally Unique Identifiers les DN peuvent changer, donc ils ne sont pas des identifiants fiables. Un UUID doit être utilisé avec LCUP. Le serveur doit fournir cet UUID en tant qu'attribut opérationnel single-value. (par exemple : entryUUID). **LCUPUUID : := OCTET STRING**

Schéma LCUP et cookie LCUP Le protocole LCUP utilise un cookie pour maintenir l'état des données client. Chaque format de cookie est identifié de manière unique avec son schéma. **LCUPScheme : := LDAPOID**. C'est l'OID qui identifie le format de la valeur du cookieLCUP. **LCUPCookie : := OCTET STRING**

contexte LCUP La partie du DIT qui est autorisé pour LCUP est appelé un contexte LCUP. Un serveur peut supporter un ou plusieurs contextes LCUP. Chaque contexte LCUP peut utiliser un schéma de cookie différent.

ResultCode additionnels

lcupResourcesExhausted (113) Le serveur n'a plus de ressources

lcupSecurityViolation (114) Le client est suspecté d'actions malicieuses

lcupInvalidData (115) Schéma ou cookie invalide

lcupUnsupportedScheme (116) Le schéma du cookie est un OID valide mais n'estpas supporté par ce serveur

lcupReloadRequired (117) Indique que les données client doivent être réinitialisées si le serveur ne contient pas suffisamment d'information pour synchroniser les données, ou si les données du serveur ont été rechargée depuis la dernière session de synchronisation.

Contrôle Sync Request

Ce contrôle a **controlType** à **1.3.6.1.1.7.1** et **controlValue**, un OCTET STRING contenant un syncRequestControlValue encodé BER :

```
syncRequestControlValue ::= SEQUENCE {
 updateType ENUMERATED {
  syncOnly (0),
  syncAndPersist (1),
  persistOnly (2) },
 sendCookieInterval [0] INTEGER OPTIONAL,
 scheme [1] LCUPScheme OPTIONAL,
 cookie [2] LCUPCookie OPTIONAL
}
```

sendCookieInterval Le serveur devrait envoyer le cookie dans la valeur du contrôle Sync Update à chaque sendCookieInterval nombre de SearchResultEntry et SearchResultReference retourné au client. Par exemple, si la valeur est 5, le serveur devrait envoyer le cookie dans la valeur du contrôle Sync Update pour toutes les 5 recherches retournés au client. Si cette valeur est absente, 0 ou inférieur à 0, le serveur choisis l'interval.

Ce contrôle est seulement applicable au message searchRequest.

Contrôle Sync Update

Ce contrôle a **controlType** à **1.3.6.1.1.7.2** et **controlValue** un OCTET STRING contenant le syncUpdateControlValue encodé BER :

```
syncUpdateControlValue ::= SEQUENCE {
 stateUpdate BOOLEAN,
 entryUUID [0] LCUPUUID OPTIONAL, -- REQUIRED for entries --
 UUIDAttribute [1] AttributeType OPTIONAL,
 entryLeftSet [2] BOOLEAN,
 persistPhase [3] BOOLEAN,
 scheme [4] LCUPScheme OPTIONAL,
 cookie [5] LCUPCookie OPTIONAL
}
```

UUIDAttribute contient le nom ou l'OID de l'attribut que le client devrait utiliser pour effectuer les recherches pour les entrées basée sur l'UUID. Le client devrait être capable de l'utiliser dans un filtre de recherche d'égalité, par exemple "(<uuid attribute>=<entry UUID value>)" et devrait être capable de l'utiliser dans la liste des attributs de requête search pour retourner ses valeurs. UUIDAttribute peut être omis si le serveur ne supporte pas les recherches sur les valeurs UUID.

Contrôle Sync Done

Ce contrôle a **controlType** à **1.3.6.1.1.7.3** et **controlValue** contenant un syncDoneValue encodé BER :

```
syncDoneValue ::= SEQUENCE {
 scheme [0] LCUPScheme OPTIONAL,
 cookie [1] LCUPCookie OPTIONAL
}
```

Requêtes LCUP search

Un client initie une synchronisation ou une session de recherche persistente avec un serveur en attachant un Sync Request à un message searchRequest. La spécification de recherche détermine le DIT, le jeu d'attributs et la quantité de données que le client veut recevoir. Le contrôle Sync Request contient la spécification de requête du client. S'il y'a une condition d'erreur, le serveur doit retourner SearchResultDone avec un resultCode approprié à l'erreur.

Synchronisation initiale et re-synchronisation complète

Les champs du contrôle Sync Request doivent être spécifiés comme suit :

updateType Doit être mis à syncOnly ou syncAndPersist

sendCookieInterval Peut être définis

scheme Peut être définis. si définis, le serveur doit utiliser ce schéma ou retourner lcupUnsupportedScheme, sinon, le serveur peut utiliser tout schéma qu'il supporte

cookie Ne doit pas être définis.

Synchronisation incrémentale et de mise à jours

Les champs du contrôle Sync Request doivent être spécifiés comme suit :

updateType Doit être à syncOnly ou syncAndPersist

sendCookieInterval Peut être définis

scheme Doit être définis

cookie Doit être définis

Le client devrait toujours utiliser le dernier cookie reçu par le serveur.

Persistent Only

Les champs de Sync Request doivent être spécifié comme suit :

updateType Doit être à persistOnly

sendCookieInterval Peut être définis

scheme Peut être définis

cookie Peut être définis, mais le serveur doit l'ignorer.

Réponses LCUP Search

Dans la réponse au client, le serveur retourne 0 ou plusieurs SearchResultEntry ou SearchResultReference, suivis par un SearchResultDone. Chaque SearchResultEntry ou SearchResultReference contient aussi un contrôle Sync Update qui décrit l'état LCUP de l'entrée retournée. SearchResultDone contient un contrôle Sync Done.

Réponses Sync Update Informational

Le serveur peut utiliser le contrôle Sync Update pour retourner des informations non liées à une entrée particulière. Il peut le faire à tout moment pour retourner un cookie au client, ou pour informer le client que la phase de sync d'une recherche syncAndPersist est complète et que la phase persist a commencé. Il peut le faire durant la phase persist même quand aucune entrée n'a changé/ Pour faire çà, il faut retourner :

- un PDU SearchResultEntry avec objectName contenant le DN de baseObject de la requête de recherche et avec un attribut list vide.
- Un contrôle Sync Update avec les champs définis comme suit :

stateUpdape Doit être à TRUE

entryUUID Devrait être l'UUID du baseObject de la requête search

entryLeftSet Doit être à FALSE

persistPhase Doit être FALSE si la recherche est dans la phase sync et TRUE si elle est dans la phase persist.

UUIDAttribute Devrait seulement être définis si c'est soit le premier résultat retourné soit l'attribut a changé

scheme Doit être définis si le cookie est définis et que le format a changé, sinon il peut être omi.

cookie Devrait être définis

Durant une requête **SyncAndPersist**, le serveur doit retourner immédiatement après que la dernière entrée de la phase sync ait été envoyée et avant que la première entrée de la phase persist ait été envoyée. Celà permet au client de savoir quand la phase sync se termine et quand la phase persist commence.

Fréquence de retour de cookie

Le champ cookie d'un contrôle Sync Update peut être définis dans tout résultat retourné, durant la phase sync et persist. Le serveur devrait retourner le cookie suffisamment souvent pour que le client resynchronise dans une période de temps raisonnable en cas de déconnections par ex. Le champ sendCookieInterval est une suggestion au serveur et il devrait respecter cette valeur.

Le champ scheme doit être définis si le cookie est présent et que le format a changé, sinon il peut être omis.

Définition d'une entrée qui entre dans le jeu de résultat

Une entrée doit être considérée rentrer dans le jeu de résultat de la recherche du client si un des conditions est rencontrée :

- Durant la phase sync pour une opération sync incrémentale, l'entrée est présent dans le jeu de résultat de la recherche mais n'était pas présente avant ; typiquement une entrée qui a été ajoutée ou déplacée dans l'annuaire.
- Durant la phase persist pour une opération de recherche persistante, l'entrée entre dans le jeu de résultat de recherche, typiquement une entrée qui a été ajoutée ou déplacée dans l'annuaire.

Définition d'une entrée qui a changé

Une entrée doit être considérée changée si un ou plusieurs de ses attributs dans la liste des attributs de la recherche ont été modifiés.

Définition d'une entrée qui a quitté le jeu de résultat

Une entrée doit être considérée sortie du jeu de résultat si elle rencontre une de ces conditions :

- Durant la phase sync, l'entrée n'est pas présente dans le jeu de résultat mais était présent avant.
- Durant la phase persist, l'entrée quitte le jeu de résultat.

Résultats pour les entrées présentes dans le jeu de résultat

Une entrée devrait être retournée présente sous les conditions suivante :

- La requête est une synchronisation initiale pour une requête full resync et l'entrée est présent dans le jeu de résultat de la recherche du client.
- La requête est une synchronisation incrémentale et l'entrée a changé ou est entrée dans le jeu de résultat depuis la dernière synchronisation.
- La recherche est dans la phase persist et l'entrée entre dans le jeu de résultat ou change.

Pour un retour SearchResultEntry, les champs du contrôle Sync Update doivent être définis comme suit :

stateUpdate Doit être FALSE

entryUUID Doit être l'UUID de l'entrée

entryLeftSet Doit être FALSE

persistPhase Doit être FALSE durant la phase sync, TRUE durant la phase Persist

UUIDAttribute Devrait être définis seulement si c'est le premier résultat retourné ou si l'attribut a changé

schema comme plus haut

cookie Comme plus haut

Résultat pour les entrées qui ont quitté le jeu de résultat

Une entrée devrait être considéré ayant quitté le jeu de résultat si elle est dans les conditions suivantes :

- La requête est une synchronisation incrémentale durant la phase sync et l'entrée est sortie du jeu de résultat
- La recherche est dans la phase persist et l'entrée a quitté le jeu de résultat
- L'entrée a quitté le jeu derésultat suitte à un LDAP delte ou ldap ModifyDN

stateUpdate Doit être FALSE

entryUUID Doit être l'UUID de l'entrée qui a quitté le jeu de résultat

entryLeftSet Doit être TRUE

persistPhase Doit être FALSE durant la phase sync, TRUE durant la phase Persist

UUIDAttribute Devrait être définis seulement si c'est le premier résultat retourné ou si l'attribut a changé

schema comme plus haut

cookie Comme plus haut

Retourner les résultats durant la phase persistante

Le serveur devrait retourner les entrées changées au client le plus rapidement possible

Pas de mixe de phase sync et persist

Durant la phase sync, le serveur ne doit pas retourner d'entrée avec le flag persistPhase à TRUE et durant la phase persisit ce flag doit être à TRUE.

Retourner les résultats mis à jours durant la phase sync

Il peut y avoir des mises à jours des entrées dans le jeu de résultats durant l'opération de recherche. Si le serveur est surchargé de mises à jours et qu'il tente d'envoyer toutes ces mises à jours, il peut ne jamais terminer la phase sync. C'est à la discretion de l'implémentation du serveur de décider quand le client est en "in sync", c'est à dire quand se termine un suncOnly ou quand envoyer la réponse Update Informational entre les phases sync et persist.

Attributs opérationnels et entrées administratives

Ils devraient être retournés tels qu'il seraient retournés dans une recherche normal. Si le serveur ne supporte pas la synchronisation des attributs opérationnel, le serveur doit retourner SearchResultDone avec un result unwillingToPerform

Garantie de convergence

Si durant une recherche LCUP soit durant la phase sync ou persist, le serveur détermine qu'il ne peut pas garantir la délivrance de la copie au client pour une éventuelle convergence, il devrait immédiatement terminer la recherche LCUP et retourner un resultCode lcupReloadRequired.

Fin de recherche LCUP

Fin initiée par le serveur Quand le serveur a terminé le traitement avec succès, il attache un contrôle Sync Done au message SearchResultDone et il l'envoie au client. Cependant, si le resultCode de ce message est différent de success ou canceled, le contrôle Sync Done peut être omis. Le cookie doit être présent si resultCode est success ou canceled. Il devrait être présent si le resultCode est lcupResourcesExhausted, timeLimitExceeded, sizeLimitExceeded, ou adminLimitExceeded. Celà permet au client de resynchroniser plus facilement.

Fin initiée par le client Si le client doit terminer la synchronisation et veut obtenir le cookie qui représente l'état courant, il fournis une opération LDAP Cancel. Le serveur répond immédiatement avec une réponse LDAP Cancel.

Limite de temps et de taille Le serveur doit supporter les limites de taille et de temps. Le serveur doit s'assurer que si l'opération est terminée dû à ces conditions, le cookie est renvoyé au client.

Opération sur la même connexion Il est permis au client de fournis des opérations LDAP sur la connexion utilisé par le protocole.

Intéraction avec d'autres contrôles LCUP ne définit pas de restriction ni ne garantie la capacité d'utiliser les contrôles de ce document avec d'autres contrôles.

Considérations de réplication L'utilisation d'un cookie LCUP avec plusieurs DSA dans un environnement répliqué n'estpas définis parLCUP. Une implémentation LCUP peut supporter la continuation de session avec un autre DSA. Les clients peuvent envoyer des cookies retournés par un DSA à un autre DSA.

Considérations côté client

Utiliser les cookies avec différents critères de recherche Le cookie reçu d'un serveur après une session de synchronisation devrait seulement être utilisé avec les même spécifications de recherche qui ont généré le cookie.

Manipulation de référrant Le client peut reçevoir un référrant quandla recherche de base est une référence subordonnées et va terminer l'opération.

Considération d'implémentation serveur

Support des UUID Les serveurs doivent supporter les UUID.

Exemple en utilisant un RUV comme valeur de cookie Par concept, le protocole supporte plusieurs schéma de cookie, ce qui permet à différentes implémentations la flexibilité du stockage des informations. Une implémentation raisonnable serait d'utiliser le ReplicaUpdate Vector. Pour chaque Master, RUV contient le plus grand CSN vu par ce master. En plus, RUV implémenté par certains serveurs contiennent une génération de réplica, un chaîne opaque qui identifie le stockage des données de réplica. La valeur change quand les données sont rechargées. La génération de réplica est utilisé pour signaler aux parties de réplication/synchronisation que les données de réplica ont été rechargée et que tous les autres réplica doivent être réinitialisées.

Support pour plusieurs schéma de cookie Un serveur doit publier les OID des schéma de cookie supportés