Groupe de travail Réseau
Request for Comments : 2743
RFC rendue obsolète : 2078
Catégorie : En cours de normalisation

J. Linn, RSA Laboratories
janvier 2000

Traduction Claude Brière de L'Isle

Interface de programme d'application pour service de sécurité générique version 2, mise à jour 1

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Normes officielles des protocoles de l'Internet" (STD 1) pour connaître l'état de la normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

RésuméL'interface de programme d'application pour service de sécurité générique (GSS-API, *Generic Security Service Application Program Interface*) version 2, telle que définie dans la [RFC2078], fournit des services de sécurité aux appelants d'une façon générique, prise en charge par une gamme de mécanismes et technologies sous-jacents permettant donc la portabilité au niveau source d'applications à différents environnements. La présente spécification définit les services et primitives de GSS-API à un niveau indépendant du mécanisme et de l'environnement de langage de programmation sous-jacent, et sera complétée par d'autres spécifications en rapport :

- documents définissant des liens de paramètres spécifiques pour des environnements de langage particuliers,
- documents définissant des formats de jetons, protocoles, et procédures à mettre en œuvre afin de réaliser des services de GSS-API par dessus des mécanismes de sécurité particuliers.

Le présent mémoire rend obsolète la [RFC2078] en faisant des changements spécifiques par incrément en réponse à l'expérience de mise en œuvre et aux demandes de liaison. Il est donc prévu que le présent mémoire ou une de ses versions ultérieures devienne la base des progrès de la spécification GSS-API sur la voie de la normalisation.

Table des Matières

1. Caractéristiques et concepts de GSS-API

GSS-API fonctionne avec le paradigme suivant. Un appelant normal de GSS-API est lui-même un protocole de communications, qui appelle sur GSS-API afin de protéger ses communications avec des services de sécurité d'authentification, d'intégrité, et/ou de confidentialité. Un appelant GSS-API accepte les jetons qui lui sont fournis par sa

mise en œuvre GSS-API locale et transfère les jetons à un homologue sur un système distant ; cet homologue passe les jetons reçus à sa mise en œuvre GSS-API locale pour traitement. Les services de sécurité disponibles à travers GSS-API de cette façon peuvent être mis en œuvre (et sont mis en œuvre) sur une gamme de mécanismes sous-jacents fondés sur les technologies de chiffrement à clé secrète et à clé publique.

GSS-API sépare les opérations d'initialisation d'un contexte de sécurité entre homologues, réalisant l'authentification d'entité homologue (invocations GSS_Init_sec_context() et GSS_Accept_sec_context()), des opérations de fourniture de l'authentification de l'origine des données et de protection de l'intégrité des données par message (invocations de GSS_GetMIC() et GSS_VerifyMIC()) pour les messages ultérieurement transférés en conjonction avec ce contexte. (La définition du service d'authentification de l'entité homologue, et les autres définitions utilisées dans ce document, correspondent à celles fournies dans [ISO-7498-2].) Lors de l'établissement d'un contexte de sécurité, GSS-API donne à l'initiateur de contexte la faculté de permettre que ses accréditifs soient délégués, ce qui signifie que l'accepteur de contexte peut initier plus de contextes de sécurité au nom de l'appelant initiateur. Les invocations de GSS_Wrap() et GSS_Unwrap() par message fournissent des services d'authentification d'origine et d'intégrité des données qu'offrent GSS_GetMIC() et GSS_VerifyMIC(), et aussi la prise en charge de la sélection des services de confidentialité comme option d'appel. Des invocations supplémentaires fournissent des fonctions de soutien aux utilisateurs de GSS-API.

Les paragraphes qui suivent donnent un exemple qui illustre les flux de données impliqués dans l'utilisation de GSS-API par un client et un serveur d'une façon indépendante du mécanisme, établissant un contexte de sécurité et transférant un message protégé. L'exemple suppose que l'acquisition d'accréditifs a déjà été achevée. L'exemple suppose aussi que la technologie d'authentification sous-jacente est capable d'authentifier un client auprès d'un serveur en utilisant des éléments portés au sein d'un seul jeton, et d'authentifier le serveur auprès du client (authentification mutuelle) avec un seul jeton retourné ; cette hypothèse tient pour certains mécanismes de CAT actuellement documentés mais n'est pas nécessairement vraie pour d'autres technologies de chiffrement et leurs protocoles associés.

Le client invoque GSS_Init_sec_context() pour établir un contexte de sécurité avec le serveur identifié par targ_name *(nom de cible)*, et choisit d'établir le fanion mutual_req_flag afin que l'authentification mutuelle soit effectuée dans le cours de l'établissement de contexte. GSS_Init_sec_context() retourne un jeton de résultat à passer au serveur, et qui indique l'achèvement en cours de l'état GSS_S_CONTINUE_NEEDED de la séquence d'authentification mutuelle. Si le fanion mutual_req_flag n'avait pas été établi, l'invocation initiale de GSS_Init_sec_context() aurait retourné l'état GSS_S_COMPLETE. Le client envoie le jeton de résultat au serveur.

Le serveur passe le jeton reçu comme paramètre de jeton d'entrée à GSS_Accept_sec_context(). GSS_Accept_sec_context indique l'état GSS_S_COMPLETE, et fournit l'identité authentifiée du client dans le résultat src_name *(nom de source)*, et fournit un jeton de résultat à passer au client. Le serveur envoie le jeton de résultat au client.

Le client passe le jeton reçu comme paramètre de jeton d'entrée d'un appel réussi à GSS_Init_sec_context(), qui traite les données incluses dans le jeton afin de réaliser l'authentification mutuelle du point de vue du client. Cette invocation de GSS_Init_sec_context() retourne l'état GSS_S_COMPLETE, qui indique la réussite de l'authentification mutuelle et l'achèvement de l'établissement du contexte pour cet exemple.

Le client génère un message de données et le passe à GSS_Wrap(). GSS_Wrap() effectue le traitement d'authentification d'origine des données, d'intégrité des données, et (facultativement) de confidentialité sur le message et encapsule le résultat dans output_message, indiquant l'état GSS_S_COMPLETE. Le client envoie le message de résultat au serveur.

Le serveur passe le message reçu à GSS_Unwrap(). GSS_Unwrap() inverse l'encapsulation effectuée par GSS_Wrap(), déchiffre le message si le dispositif facultatif de confidentialité a été appliqué, et valide les quantités d'authentification d'origine des données et de vérification d'intégrité des données. GSS_Unwrap() indique la réussite de la validation en retournant l'état GSS_S_COMPLETE avec le message de résultat résultant.

Pour les besoins de cet exemple, on suppose que le serveur sait par des moyens hors bande que ce contexte n'aura pas d'autre utilité après qu'un message protégé est transféré du client au serveur. Ces prémices étant posées, le serveur invoque maintenant GSS_Delete_sec_context() pour purger les informations de niveau contexte. Facultativement, l'application côté serveur peut fournir une mémoire tampon de jeton à GSS_Delete_sec_context(), pour recevoir un jeton de contexte à transférer au client afin de demander que les informations de niveau contexte du côté client soient supprimées.

Si un jeton de contexte est transféré, le client passe le jeton de contexte à GSS_Process_context_token(), qui retourne un état GSS_S_COMPLETE après avoir supprimé les informations de niveau contexte au système client.

Le concept de GSS-API suppose et vise plusieurs buts de base , incluant :

L'indépendance au mécanisme : GSS-API définit une interface aux services d'authentification forte mise en œuvre par chiffrement et autres services de sécurité à un niveau générique qui est indépendant des mécanismes particuliers sous-

jacents. Par exemple, les services fournis par GSS-API ont été mis en œuvre en utilisant les technologies de clé secrète (par exemple, Kerberos, selon la [RFC1964]) et les approches à clé publique (par exemple, SPKM, selon la [RFC2025]).

L'indépendance à l'environnement de protocole : GSS-API est indépendant des suites de protocoles de communications avec lesquelles il est employé, ce qui permet de l'utiliser dans une large gamme d'environnements de protocoles. Dans les environnements appropriés, un "plaquage" de mise en œuvre intermédiaire orientée vers un protocole de communication particulier peut être interposé entre les applications qui invoquent ce protocole et GSS-API (par exemple, comme défini dans la [RFC2203] pour l'invocation de procédure distante (RPC, *Remote Procedure Call*) de calcul sur réseau ouvert, invoquant par là les facilités de GSS-API en conjonction avec les invocations de communications de ce protocole.

L'indépendance d'association de protocole : La construction du contexte de sécurité de GSS-API est indépendante des constructions d'association de protocole de communications. Cette caractéristique permet à une seule mise en œuvre de GSS-API d'être utilisée par divers modules de protocoles invocateurs au nom des applications appelantes de ces modules. Les services GSS-API peuvent aussi être invoqués directement par les applications, en complète indépendance des associations de protocole.

Convenable pour une gamme de placements de mise en œuvre : Les clients GSS-API ne sont pas obligés de résider au sein d'un périmètre de base de calcul de confiance (TCB, *Trusted Computing Base*) définie sur un système où GSS-API est mis en œuvre ; les services de sécurité sont spécifiés d'une manière qui convient aux appelants aussi bien intra-TCB que extra-TCB.

1.1 Constructions de GSS-API

Cette section décrit les éléments de base qui composent la GSS-API.

1.1.1 Accréditifs

1.1.1.1 Constructions et concepts d'accréditifs

Les accréditifs fournissent les prérequis qui permettent aux homologues GSS-API d'établir des contextes de sécurité les uns avec les autres. Un appelant peut déclarer que les éléments d'accréditifs qui sont à appliquer pour l'initiation ou l'acceptation de contexte sont choisis par défaut. Autrement, les appelants GSS-API qui ont besoin de faire un choix explicite de structures d'accréditifs particulières peuvent faire référence à ces accréditifs à travers les liens d'accréditifs ("cred_handles") fournis par GSS-API. Dans tous les cas, les références d'accréditifs des appelants sont indirectes, médiatisées par les mises en œuvre de GSS-API et n'exigent pas des appelants qu'ils accèdent aux éléments d'accréditifs choisis.

Une seule structure d'accréditif peut être utilisée pour initier les contextes de sortie et pour accepter les contextes d'entrée. Les appelants qui ont besoin de fonctionner dans un seul de ces modes peuvent mentionner ce fait lorsque les accréditifs sont acquis pour être utilisés, ce qui permet aux mécanismes sous-jacents d'optimiser leurs exigences de traitement et de mémorisation. Les éléments d'accréditifs définis par un mécanisme particulier peuvent contenir plusieurs clés de chiffrement, par exemple, pour permettre d'effectuer avec des algorithmes différents l'authentification et le chiffrement de message.

Une structure d'accréditif GSS-API peut contenir plusieurs éléments d'accréditifs, contenant chacun des informations spécifiques d'un mécanisme pour un type de mécanisme (mech_type) sous-jacent particulier, mais l'ensemble des éléments au sein d'une certaine structure d'accréditifs représente une entité commune. Le contenu d'une structure d'accréditif va varier selon l'ensemble de mech_types pris en charge par une mise en œuvre GSS-API particulière. Chaque élément d'accréditif identifie les données nécessaires pour ce mécanisme afin d'établir les contextes au nom d'un principal particulier, et peut contenir des références d'accréditif distinctes pour les utiliser dans l'initialisation et l'acceptation de contexte. Plusieurs éléments d'accréditif au sein d'un certain accréditif ayant des combinaisons de mécanisme, de mode d'usage, et de période de validité qui se chevauchent, ne sont pas permises.

Normalement, un seul mech_type sera utilisé pour tous les contextes de sécurité établis par un initiateur particulier avec une cible particulière. Un motif majeur de la prise en charge d'ensembles d'accréditifs représentant plusieurs types de mécanismes est de permettre aux initiateurs sur des systèmes qui sont équipés pour traiter plusieurs types d'initier les contextes avec des cibles sur d'autres systèmes qui ne peuvent s'accommoder que d'un sous-ensemble de celui accepté par le système de l'initiateur.

1.1.1.2 Gestion d'accréditifs

Il est de la responsabilité des mécanismes spécifiques du système sous-jacent et des fonctions de système d'exploitation en dessous de GSS-API de s'assurer que la capacité à acquérir et utiliser les accréditifs associés à une certaine identité est limitée aux processus appropriés au sein d'un système. Cette responsabilité devrait être prise au sérieux par les mises en œuvre car la capacité d'une entité à utiliser les accréditifs d'un principal est équivalente à la capacité de cette entité à

certifier avec succès l'identité de ce principal.

Une fois qu'est établi un ensemble d'accréditifs de GSS-API, la capacité à transférer cet ensemble d'accréditifs aux autres processus ou constructions analogues au sein d'un système est une affaire locale, non définie par la GSS-API. Un exemple de politique locale serait celui où tout accréditif reçu par suite d'une connexion sur un certain compte d'usager, ou d'une délégation de droits à ce compte, serait accessible par, ou transférable à, des processus fonctionnant sur ce compte.

Le processus d'établissement d'accréditif (en particulier lorsque il est effectué au nom des usagers plutôt que des processus de serveur) va probablement exiger un accès à des mots de passe ou autres quantités qui devraient être protégées en local et exposées pendant le plus court instant possible. Par suite, il sera souvent approprié que l'établissement d'accréditifs préliminaires soit effectué par des moyens locaux au moment de l'amorçage de l'utilisateur, le ou les résultats étant placés en antémémoire pour référence ultérieure. Ces accréditifs préliminaires seraient mis de côté (d'une façon spécifique du système) pour utilisation ultérieure, soit :

- pour y accéder par une invocation de GSS_Acquire_cred(), retournant un lien explicite pour référence de cet accréditif,
- pour comprendre les éléments d'accréditif par défaut à installer, et à utiliser lorsque le comportement d'accréditif par défaut est demandé au titre d'un processus.

1.1.1.3 Résolution par défaut d'accréditifs

Les sous-programmes GSS_Init_sec_context() et GSS_Accept_sec_context() permettent de spécifier la valeur GSS_C_NO_CREDENTIAL comme paramètre de lien d'accréditif. Ce lien spécial d'accréditif indique le désir de l'application d'agir comme principal par défaut. En soutien de la portabilité d'application, la prise en charge du comportement de résolution par défaut décrit ci-dessous pour les accréditifs d'initiateur (l'usage GSS_Init_sec_context()) est obligatoire ; la prise en charge du comportement de résolution par défaut décrit ci-dessous pour l'accepteur des accréditifs (l'usage GSS_Accept_sec_context()) est recommandée. Si la résolution d'accréditif par défaut échoue, on doit retourner l'état GSS_S_NO_CRED.

GSS_Init_sec_context :

(i) s'il n'y a qu'un seul principal capable d'initier les contextes de sécurité au nom desquels l'application est autorisée à agir, ce principal devra être utilisé ; autrement,

(ii) si la plateforme entretient un concept d'identité réseau par défaut, et si l'application est autorisée à agir au nom de cette identité pour les besoins de l'initiation des contextes de sécurité, alors le principal correspondant à cette identité devra être utilisée ; autrement,

(iii) si la plateforme entretient un concept d'identité locales par défaut, et fournit un moyen pour transposer les identités locales en identités réseau, et si l'application est autorisée à agir au nom de l'image d'identité réseau de l'identité locale par défaut pour les besoins de l'initiation des contextes de sécurité, le principal correspondant à cette identité devra alors être utilisé ; autrement,

(iv) une identité par défaut configurable par l'usager devrait être utilisée.

GSS_Accept_sec_context :

(i) si il y a une seule identité de principal autorisée capable d'accepter les contextes de sécurité, ce principal devra alors être utilisé ; autrement,

(ii) si le mécanisme peut déterminer l'identité du principal cible en examinant le jeton d'établissement de contexte, et si l'application acceptante est autorisée à agir comme ce principal pour les besoins d'acceptation des contextes de sécurité, cette identité de principal devra alors être utilisée ; autrement,

(iii) si le mécanisme prend en charge l'acceptation de contexte par tout principal, et si l'authentification mutuelle n'était pas exigée, on peut utiliser tout principal que l'application est autorisée à accepter sous les contextes de sécurité ; autrement,

(iv) on devra utiliser une identité par défaut configurable par l'usager.

L'objet des règles ci-dessus est de permettre aux contextes de sécurité d'être établis par les deux initiateur et accepteur en utilisant le comportement par défaut chaque fois que possible. Les applications qui demandent le comportement par défaut seront vraisemblablement plus portables à travers les mécanismes et plateformes que ceux qui utilisent GSS_Acquire_cred() pour demander une identité spécifique.

1.1.2 Jetons

Les jetons sont des éléments de données transférés entre les appelants GSS-API, et sont divisés en deux classes. Les jetons de niveau contexte sont échangés afin d'établir et gérer un contexte de sécurité entre les homologues. Les jetons par message se rapportent à un contexte établi et sont échangés pour fournir des services de sécurité de protection (c'est-à-dire, l'authentification d'origine des données, l'intégrité, et facultativement la confidentialité) pour les messages de données correspondants.

Le premier jeton de niveau contexte obtenu de GSS_Init_sec_context() est exigé pour indiquer à son tout début un identifiant de mécanisme interprétable mondialement, c'est-à-dire, un identifiant d'objet (OID, *Object Identifier*) du mécanisme de sécurité. La partie restante de ce jeton ainsi que tout le contenu de tous les autres jetons est spécifique du mécanisme sous-jacent particulier utilisé pour la prise en charge de la GSS-API. Le paragraphe 3.1 de ce document fournit, pour les concepteurs de mécanismes de GSS-API, la description de l'en-tête du premier jeton de niveau contexte qui est alors suivi des informations spécifiques du mécanisme.

Les contenus de jetons sont opaques du point de vue des appelants de GSS-API. Ils sont générés au sein de la mise en œuvre de GSS-API au système d'extrémité, fournis à un appelant GSS-API pour être transférés à l'appelant homologue GSS-API au système d'extrémité distant, et traités par la mise en œuvre GSS-API au système d'extrémité distant.

Les jetons de niveau contexte peuvent être produits par les appels GSS-API (et devraient être transférés aux homologues GSS-API) que les indicateurs d'état des appels indiquent ou non l'achèvement réussi. À l'opposé, les jetons par message ne sont à retourner qu'à l'achèvement réussi des appels par message. Les jetons de longueur zéro ne sont jamais retournés par les sous-programmes GSS pour le transfert à un homologue. Le transfert de jeton peut avoir lieu dans la bande, intégré dans le même flux de protocole utilisé par les appelants GSS-API pour d'autres transferts de données, ou hors bande à travers un canal logiquement séparé.

Des jetons GSS-API différents sont utilisés pour différents objets (par exemple, initialisation de contexte, acceptation de contexte, données de message protégées sur un conteste établi) et il est de la responsabilité d'un appelant GSS-API qui reçoit des jetons de distinguer leurs types, de les associer aux contextes de sécurité correspondants, et de les passer aux sous-programmes de traitement GSS-API appropriés. Selon l'environnement de protocole de l'appelant, cette distinction peut être réalisée de plusieurs façons.

Les exemples suivants illustrent les moyens par lesquels les types de jetons peuvent être distingués :
- étiquetage implicite fondé sur les informations d'état (par exemple, tous les jetons sur une nouvelle association sont considérés comme étant des jetons d'établissement de contexte jusqu'à ce que l'établissement du contexte soit terminé, moment auquel tous les jetons sont considérés comme étant des objets de données enveloppées pour ce contexte),
- étiquetage explicite au niveau du protocole d'appelant,
- un mélange de ces approches.

Le plus souvent, les données encapsulées au sein d'un jeton incluent des informations d'étiquetage spécifiques du mécanisme interne qui permettent à des modules de traitement au niveau du mécanisme de distinguer les jetons utilisés au sein du mécanisme pour les différents objets. Un tel étiquetage au niveau du mécanisme interne est recommandé aux concepteurs de mécanismes, et permet aux mécanismes de déterminer si un appelant a passé un jeton particulier pour le traitement par un sous-programme GSS-API inapproprié.

Le développement d'un mécanisme GSS-API fondé sur une technique et protocole de chiffrement sous-jacent particulier (c'est-à-dire, conforme à une définition de mécanisme GSS-AOI spécifique) n'implique pas nécessairement que les appelants GSS-API qui utilisent ce mécanisme de GSS-API seront capables d'interopérer avec les homologues qui invoquent les mêmes technique et protocole en-dehors du paradigme GSS-API, ou avec des homologues qui mettent en œuvre un mécanisme GSS-API différent fondé sur la même technologie sous-jacente. Le format des jetons GSS-API défini en conjonction avec un mécanisme particulier, et les techniques utilisées pour intégrer ces jetons dans les protocoles des appelants, peut n'être pas interopérable avec les jetons utilisés par des appelants non GSS-API de la même technique sous-jacente.

1.1.3 Contextes de sécurité

Les contextes de sécurité sont établis entre les homologues, en utilisant les accréditifs établis localement en conjonction avec chaque homologue ou reçus par les homologues via une délégation. Plusieurs contextes peuvent exister simultanément entre une paire d'homologues, en utilisant le même ensemble d'accréditifs ou des ensembles différents. La coexistence de plusieurs contextes utilisant des accréditifs différents permet un débordement en douceur lorsque les accréditifs arrivent à expiration. La distinction parmi plusieurs contextes fondés sur les mêmes accréditifs sert aux applications à distinguer les différents flux de messages au sens de la sécurité.

GSS-API est indépendant des protocoles sous-jacents et de la structure d'adressage, et dépend de ses appelants pour transporter les éléments de données fournis par GSS-API. Il résulte de ces facteurs qu'il est de la responsabilité de l'appelant d'analyser les messages communiqués, en séparant les éléments de données en rapport avec GSS-API des données fournies par l'appelant. GSS-API est indépendant de l'orientation connexion/sans connexion du service de communications sous-jacent.

Il n'y a pas de corrélation entre contexte de sécurité et association de protocole de communications. (La facilité facultative

de liaison de canal, discutée au paragraphe 1.1.6 de ce document, représente une exception intentionnelle à cette règle, prenant en charge des dispositifs de protection supplémentaires au sein des mécanismes de prise en charge de GSS-API.) Cette séparation permet à GSS-API d'être utilisé dans une large gamme d'environnements de communications, et aussi simplifie les séquences d'appels des appels individuels. Dans de nombreux cas (selon le protocole de sécurité sous-jacent, les mécanismes associés, et la disponibilité des informations en antémémoire) les informations d'état requises pour l'établissement de contexte peuvent être envoyées concurremment avec les données initiales signées d'utilisateur, sans interposer d'échanges de messages supplémentaires. Les messages peuvent être protégés et transférés concurremment dans les deux directions sur un contexte de sécurité GSS-API établi ; la protection des messages dans une direction n'interfère pas avec la protection des messages dans la direction inverse.

Les mises en œuvre de GSS-API sont supposées conserver les données de contexte interrogeables sur un contexte jusqu'à ce que celui-ci soit libéré par un appelant, même après que le contexte est arrivé à expiration, bien que les éléments de données cryptographiques sous-jacents puissent être supprimés après expiration afin de limiter leur exposition.

1.1.4 Types de mécanisme

Afin de réussir à établir un contexte de sécurité avec un homologue cible, il est nécessaire d'identifier un type de mécanisme (mech_type) sous-jacent approprié avec le soutien des deux homologues initiateur et cible. La définition d'un mécanisme incorpore non seulement l'utilisation d'une technologie de chiffrement particulière (ou un hybride de choix parmi des technologies de chiffrement de remplacement) mais aussi la définition de la syntaxe et de la sémantique des échanges d'éléments de données que ce mécanisme va employer afin de prendre en charge les services de sécurité.

Il est recommandé que les appelants qui initient des contextes spécifient la valeur de mech_type "par défaut", pour permettre aux fonctions spécifiques du système au sein de la mise en œuvre de GSS-API, ou invoquées par elle, de choisir le type de mécanisme approprié, mais les appelants peuvent décider qu'un mech_type particulier soit employé lorsque nécessaire.

Pour les besoins de GSS-API, la phrase "mécanisme de négociation" se réfère à un mécanisme qui effectue lui-même la négociation afin de choisir un mécanisme concret qui est partagé entre les homologues et est ensuite utilisé pour l'établissement du contexte. Seuls les mécanismes qui sont définis dans leur spécification comme des mécanismes de négociation vont donner les mécanismes choisis avec des valeurs d'identifiant différentes de la valeur qui est entrée par un appelant GSS-API, sauf dans le cas d'un appelant qui demande le mech_type "par défaut".

Les moyens pour identifier un mech_type partagé pour établir un contexte de sécurité avec un homologue vont varier selon les différents environnements et circonstances ; les exemples incluent (mais ne s'y limitent pas) :
- l'utilisation d'un mech_type fixé, défini par configuration, au sein d'un environnement ;
- une convention syntaxique spécifique d'une cible, par l'examen d'une recherche du nom de la cible dans un service de noms ou autre base de données afin d'identifier les types de mécanismes pris en charge par cette négociation explicite de cible entre appelants GSS-API avant l'établissement du contexte de sécurité ;
- l'utilisation d'un mécanisme de négociation.

Lorsque ils sont transférés entre les homologues GSS-API, les spécificateurs de mech_type (selon la Section 3 du présent document) représentés par des identifiants d'objet (OID)) servent à qualifier l'interprétation des jetons associés. (La structure et le codage des identifiants d'objets est définie dans [ISOIEC- 8824] et [ISOIEC-8825].) L'utilisation d'OID structurés hiérarchiquement sert à empêcher des interprétations ambiguës des spécificateurs de mech_type. L'OID qui représente, par exemple, le type de mécanisme DASS ([RFC1507]) est 1.3.12.2.1011.7.5, et celui du mécanisme Kerberos V5 ([RFC1964]) qui a été avancé au niveau de proposition de norme est 1.2.840.113554.1.2.2.

1.1.5 Dénomination

GSS-API évite de prescrire des structures de dénomination, et traite les noms qui sont transférés à travers l'interface pour initier et accepter les contextes de sécurité comme des objets opaques. Cette approche soutient le but de GSS-API de mise en œuvre par dessus une gamme de mécanismes de sécurité sous-jacents, reconnaissant le fait que différents mécanismes traitent et authentifient les noms qui sont présentés sous des formes différentes. Les services généralisés qui offrent des fonctions de traduction entre des ensembles arbitraires d'environnements de désignation sortent du domaine d'application de GSS-API ; la disponibilité et l'utilisation de fonctions locales de conversion pour traduire les formats de désignation pris en charge au sein d'un certain système d'extrémité est prévue.

Différentes classes de représentations de noms sont utilisées en conjonction avec les différents paramètres GSS-API :
- La forme interne (notée NOM INTERNE dans ce document) opaque aux appelants et définie par les mises en œuvre GSS-API individuelles. Les mises en œuvre GSS-API qui prennent en charge plusieurs types d'espace de nom doivent tenir des étiquettes internes pour rendre sans ambiguïté l'interprétation des noms. Un nom de mécanisme (MN,

Mechanism Name) est un cas particulier de NOM INTERNE, dont il est garanti qu'il contient les éléments qui correspondent à un mécanisme et un seul ; les appels dont il est garanti qu'ils émettent des MN ou qui exigent des MN en entrée sont identifiés comme tels dans la présente spécification.

- La forme de chaîne contiguë ("plate") (notée CHAINE D'OCTETS dans ce document) ; accompagnée par les étiquettes d'OID qui identifient l'espace de noms auquel elles correspondent. Selon la valeur de l'étiquette, les noms plats peuvent ou non être des chaînes imprimables pour acceptation directe et présentation aux usagers. L'étiquetage des noms plats permet aux appelants GSS-API et aux mécanismes GSS-API sous-jacents d'ôter toute ambiguïté aux types de noms et de déterminer si un type de nom associé est un de ceux qu'ils sont capables de traiter, évitant les problèmes d'alias qui pourraient résulter de la mauvaise interprétation du nom d'un type comme nom d'un autre type.
- L'objet Nom exporté GSS-API est un cas particulier de nom plat désigné par une valeur réservée d'OID ; il porte une forme de nom canonisée qui convient pour les comparaisons binaires.

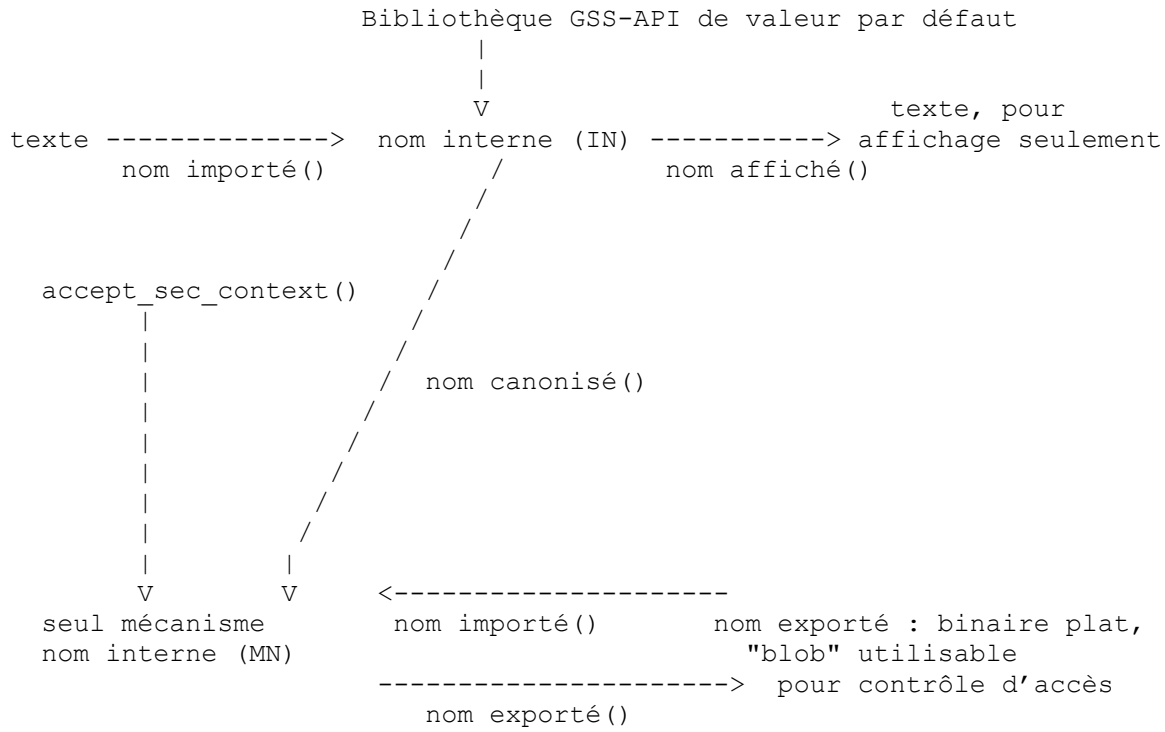
En plus de fournir le moyen d'étiqueter les noms avec les types, la présente spécification définit les primitives pour prendre en charge un niveau de désignation de l'indépendance à l'environnement pour certaines applications appelantes. Pour fournir des services de base orientés vers les exigences des appelants qui n'ont pas eux-mêmes besoin d'interpréter la syntaxe et la sémantique interne des noms, GSS-API invoque des fonctions de comparaison des noms (GSS_Compare_name()), d'affichage lisible par l'homme (GSS_Display_name()), de conversion des entrées (GSS_Import_name()), de désallocation du nom interne (GSS_Release_name()), et de duplication du nom interne (GSS_Duplicate_name()). (On prévoit que ces invocations proposées de GSS-API seront mises en œuvre dans de nombreux systèmes d'extrémité sur la base de primitives de manipulation de nom spécifiques du système déjà existantes au sein de ces systèmes d'extrémité ; l'inclusion dans GSS-API est destinée à offrir aux appelants GSS-API un moyen portable d'effectuer des opérations spécifiques, en soutien des exigences d'autorisation et d'audit, sur les noms authentifiés.)

Les mises en œuvre de GSS_Import_name() peuvent, lorsque c'est approprié, prendre en charge plus d'une syntaxe imprimable correspondant à un certain espace de noms (par exemple, d'autres représentations imprimables pour les noms distinctifs X.500) donnant de la souplesse pour que les appelants choisissent parmi les diverses représentations. Les mises en œuvre de GSS_Display_name() sortent une syntaxe imprimable choisie de façon appropriée aux environnements de fonctionnement ; ce choix est une affaire locale. Les appelants qui désirent la portabilité à travers diverses syntaxes imprimables devraient s'abstenir de mettre en œuvre des comparaisons fondées sur les formes de noms imprimables et devraient plutôt utiliser l'invocation de GSS_Compare_name() pour déterminer si un nom de format interne correspond ou nom à un autre.

Lorsque elle est utilisée dans de grandes listes de contrôle d'accès (ACL, *access control list*) la redondance de l'invocation de GSS_Import_name() et GSS_Compare_name() sur chaque nom de l'ACL peut être prohibitive. Comme autre moyen de prendre ce cas en charge, GSS-API définit une forme spéciale du nom de chaîne contiguë qui peut être comparée directement (par exemple, avec memcmp()). Les noms contigus qui conviennent pour la comparaison sont générés par le sous-programme GSS_Export_name() qui exige un MN en entrée. Les noms exportés peuvent être réimportés par le sous-programme GSS_Import_name(), et le nom interne résultant sera aussi un MN. La constante symbolique GSS_C_NT_EXPORT_NAME identifie le type "nom exporté". Structurellement, un objet Nom exporté consiste en un en-tête contenant un OID qui identifie le mécanisme ayant authentifié le nom, et un en-queue contenant le nom lui-même, où la syntaxe de l'en-queue est définie par la spécification du mécanisme individuel. Le format précis d'un nom exporté est défini au paragraphe 3.2 de cette spécification.

Noter que le résultat de l'utilisation de GSS_Compare_name() sera en général différent de celui obtenu en invoquant GSS_Canonicalize_name() et GSS_Export_name(), et en comparant ensuite les noms exportés. La première série d'opérations détermine si deux noms (non authentifiés) identifient le même principal ; la seconde si un mécanisme particulier les authentifieraient comme le même principal. Ces deux opérations ne vont en général donner le même résultat que pour les MN.

Le diagramme suivant illustre les flux de données voulus entre les sous-programmes de traitement de GSS-API en rapport avec le nom.



1.1.6 Liens de canaux

GSS-API traite le concept d'informations de lien de canal ("chan_binding") fourni par l'appelant. Les liens de canaux sont utilisés pour renforcer la qualité avec laquelle est fournie l'authentification de l'entité homologue durant l'établissement de contexte, en limitant la portée de la réutilisation d'un jeton d'établissement de contexte intercepté par un attaquant. Précisément, ils permettent aux appelants GSS-API de lier l'établissement d'un contexte de sécurité aux caractéristiques pertinentes (par exemple, aux adresses, aux représentations transformées de clés de chiffrement) du canal de communications sous-jacent, aux mécanismes de protection appliqués à ce canal de communications, et aux données spécifiques d'application.

L'appelant qui initie un contexte de sécurité doit déterminer les valeurs de lien de canal appropriées pour fournir en entrée l'invocation de GSS_Init_sec_context(), et des valeurs cohérentes doivent être fournies à GSS_Accept_sec_context() par la cible du contexte, afin que les deux mécanismes GSS-API des homologues valident que les jetons reçus possèdent les caractéristiques correctes en rapport avec le canal. L'utilisation ou non de la facilité de lien de canal GSS-API est une option de l'appelant. Les mécanismes de GSS-API peuvent fonctionner dans un environnement où des liens de canal NUL sont présentés ; les mises en œuvre de mécanisme sont invitées, mais sans y être obligées, à utiliser les données de lien de canal fournies par l'appelant au sein de leurs mécanismes. Les appelants ne devraient pas supposer que les mécanismes sous-jacents fournissent la protection de la confidentialité pour les informations de lien de canal.

Lorsque des liens de canal non NUL sont fournis par les appelants, certains mécanismes peuvent offrir une valeur de sécurité améliorée en interprétant le contenu des liens (plutôt que de simplement représenter ces liens, ou des valeurs de vérification d'intégrité calculées sur eux, au sein des jetons) et vont donc dépendre de la présentation de données spécifiques dans un format défini. À cette fin, des accords entre les mises en œuvre de mécanisme définissent des interprétations conventionnelles pour le contenu des arguments de liens de canal, incluant des spécificateurs d'adresse (avec un contenu dépendant de l'environnement de protocole de communications) pour les initiateurs et accepteurs de contexte. (Ces conventions sont incorporées dans les spécifications de mécanisme GSS-API et dans la spécification de liens de langage GSS-API C.) Pour que les appelants GSS-API soient portables sur plusieurs mécanismes et réalisent la pleine fonctionnalité de sécurité que chaque mécanisme peut fournir, il est fortement recommandé que les appelants GSS-API fournissent des liens de canal cohérents avec ces conventions et celles de l'environnement de réseautage dans lequel elles opèrent.

1.2 Caractéristiques et problèmes de GSS-API

Cette section décrit les aspects des opérations GSS-API, des services de sécurité que fournit GSS-API, et fait des commentaires sur les questions de conception.

1.2.1 Rapports d'état et prise en charge de service facultatif

1.2.1.1 Rapport d'état

Chaque appel GSS-API fournit des valeurs de retour d'état. Les valeurs d'état majeur *(Major_status)* fournissent une indication d'état d'appel indépendante du mécanisme (par exemple, GSS_S_COMPLETE, GSS_S_FAILURE, GSS_S_CONTINUE_NEEDED) suffisante pour piloter de façon générique un flux de contrôle normal au sein de l'appel. Le Tableau 1 résume les codes d'état majeur définis.

Les codes d'état majeur informatifs en rapport avec le séquençage (GSS_S_DUPLICATE_TOKEN, GSS_S_OLD_TOKEN, GSS_S_UNSEQ_TOKEN, et GSS_S_GAP_TOKEN) peuvent être indiqués en conjonction avec l'état GSS_S_COMPLETE, ou GSS_S_FAILURE pour les appels GSS-API par message. Pour les appels d'établissement de contexte, ces codes en rapport avec le séquençage ne seront indiqués qu'en conjonction avec l'état GSS_S_FAILURE (jamais en conjonction avec GSS_S_COMPLETE ou GSS_S_CONTINUE_NEEDED) et donc, ils correspondent toujours à des défaillances fatales si il s'en rencontre durant la phase d'établissement de contexte.

Tableau 1 : Codes d'état majeur de GSS-API

Codes d'erreur fatale	**Signification**
GSS_S_BAD_BINDINGS	discordance de lien de canal
GSS_S_BAD_MECH	mécanisme demandé non accepté
GSS_S_BAD_NAME	nom fourni invalide
GSS_S_BAD_NAMETYPE	le nom fourni est d'un type non pris en charge
GSS_S_BAD_STATUS	sélecteur d'état d'entrée invalide
GSS_S_BAD_SIG	vérification d'intégrité du jeton invalide
GSS_S_BAD_MIC	alias préféré pour GSS_S_BAD_SIG
GSS_S_CONTEXT_EXPIRED	contexte de sécurité spécifié expiré
GSS_S_CREDENTIALS_EXPIRED	accréditif détecté expiré
GSS_S_DEFECTIVE_CREDENTIAL	accréditif détecté défectueux
GSS_S_DEFECTIVE_TOKEN	jeton détecté défectueux
GSS_S_FAILURE	défaillance, non spécifiée au niveau GSS-API
GSS_S_NO_CONTEXT	pas de contexte de sécurité valide spécifié
GSS_S_NO_CRED	pas d'accréditif valide fourni
GSS_S_BAD_QOP	valeur de QOP non prise en charge
GSS_S_UNAUTHORIZED	opération non autorisée
GSS_S_UNAVAILABLE	opération non disponible
GSS_S_DUPLICATE_ELEMENT	élément d'accréditif dupliqué demandé
GSS_S_NAME_NOT_MN	nom contenant des éléments multi mécanismes

Codes d'erreur pour information	**Signification**
GSS_S_COMPLETE	achèvement normal
GSS_S_CONTINUE_NEEDED	continuation de l'invocation du sous-programme exigée
GSS_S_DUPLICATE_TOKEN	détection d'un jeton par message dupliqué
GSS_S_OLD_TOKEN	jeton par message périmé détecté
GSS_S_UNSEQ_TOKEN	jeton par message déclassé (trop tôt) détecté
GSS_S_GAP_TOKEN	jetons antérieurs sautés détectés

Les états mineurs *(Minor_status)* fournissent des informations d'état plus détaillées qui peuvent inclure des codes d'état spécifiques du mécanisme de sécurité sous-jacent. Les valeurs d'état mineur ne sont pas spécifiée dans ce document.

Les retours d'état majeur GSS_S_CONTINUE_NEEDED, et les résultats de message facultatifs, sont fournis dans les invocations GSS_Init_sec_context() et GSS_Accept_sec_context() de sorte que l'emploi de différents nombres de messages de différents mécanismes au sein de leurs séquences d'authentification n'a pas besoin d'être reflété dans des chemins de code séparés au sein des applications appelantes. De tels cas sont plutôt traités avec des séquences de continuation d'invocation à GSS_Init_sec_context() et GSS_Accept_sec_context(). La même facilité est utilisée pour encapsuler l'authentification mutuelle au sein des appels d'initiation de contexte de GSS-API.

Pour les types de mécanisme *(mech_type)* qui exigent des interactions avec des serveurs tiers afin d'établir un contexte de sécurité, les appels d'établissement de contexte GSS-API peuvent bloquer l'achèvement en cours de telles interactions de tiers. D'un autre côté, aucun appel GSS-API n'attend d'interactions en série avec des entités GSS-API homologues. Il en résulte que les retours d'état local GSS-API ne peuvent pas refléter des exceptions imprévisibles ou asynchrones survenant chez les homologues distants, et refléter de telles informations d'état est une responsabilité de l'appelant qui sort du domaine d'application de GSS-API.

1.2.1.2 Prise en charge de service facultatif

Un initiateur de contexte peut demander divers services facultatifs au moment de l'établissement de contexte. Chacun de ces services est demandé en établissant un fanion dans le paramètre d'entrée req_flags pour GSS_Init_sec_context().

Les services facultatifs actuellement définis sont :

- Délégation - Le transfert de droits (habituellement temporaire) de l'initiateur à l'accepteur, qui permet à l'accepteur de s'authentifier comme agent de l'initiateur.

- Authentification mutuelle - En plus de l'authentification par l'initiateur de son identité auprès de l'accepteur de contexte, celui ci devrait aussi s'authentifier auprès de l'initiateur.

- Détection de répétition - En plus de la fourniture des services d'intégrité du message, GSS_GetMIC() et GSS_Wrap() devraient inclure des informations de numérotation de message pour permettre à GSS_VerifyMIC() et GSS_Unwrap() de détecter si un message a été dupliqué.

- Détection des hors séquence - En plus de la fourniture des services d'intégrité du message, GSS_GetMIC() et GSS_Wrap() devraient inclure des informations de numérotation de message pour permettre à GSS_VerifyMIC() et GSS_Unwrap() de détecter si un message a été reçu hors séquence.

- Authentification anonyme – L'établissement du contexte de sécurité ne devrait pas révéler l'identité de l'initiateur à l'accepteur de contexte.

- Confidentialité disponible par message - Demandes que les services de confidentialité par message soient disponibles sur le contexte.

- Intégrité disponible par message - Demandes que les services d'intégrité par message soient disponibles sur le contexte.

Tout bit non défini actuellement dans ces arguments de fanion devrait être ignoré par les mises en œuvre de GSS-API lors de la présentation par l'application, et devrait être mis à zéro au retour à l'application par la mise en œuvre GSS-API.

Certains mécanismes peuvent ne pas prendre en charge tous les services facultatifs, et certains mécanismes peuvent ne prendre en charge que certains services en conjonction avec d'autres. GSS_Init_sec_context() et GSS_Accept_sec_context() informent tous deux les applications des services qui seront disponibles à partir du contexte lorsque la phase d'établissement est terminée, via le paramètre de sortie ret_flags *(fanions de retour)*. En général, si le mécanisme de sécurité est capable de fournir un service demandé, il devrait le faire, même si des services supplémentaires doivent être activés afin de fournir le service demandé. Si le mécanisme est incapable de fournir un service demandé, il devrait continuer sans le service, laissant l'application interrompre le processus d'établissement de contexte si elle considère que le service demandé est obligatoire.

Certains mécanismes peuvent spécifier que la prise en charge de certains services est facultative, et que les mises en œuvre du mécanisme n'ont pas besoin de le fournir. Ceci est d'autant plus vrai du service de confidentialité, souvent à cause de restrictions légales sur l'utilisation du chiffrement des données, mais peut s'appliquer à n'importe lequel des services. Il est exigé de ces mécanismes qu'ils envoient au moins un jeton de l'accepteur à l'initiateur durant l'établissement de contexte Lorsque l'initiateur indique son désir d'utiliser un tel service, afin que l'initiateur GSS-API puisse correctement indiquer si le service est pris en charge par la GSS-API de l'accepteur.

1.2.2 Disponibilité de service de sécurité par message

Lorsque un contexte est établi, deux fanions sont retournés pour indiquer l'ensemble de services de sécurité de protection par message qui va être disponible sur le contexte :
- le fanion integ_avail qui indique si les services d'intégrité et d'authentification d'origine des données par message sont disponibles ;
- le fanion conf_avail qui indique si les services de confidentialité par message sont disponibles, et ne sera jamais retourné VRAI sauf si le fanion integ_avail est aussi retourné VRAI.

Les appelants GSS-API qui désirent des services de sécurité par message devraient vérifier les valeurs de ces fanions au moment de l'établissement de contexte, et doivent savoir qu'une valeur FAUX retournée pour integ_avail signifie que l'invocation des primitives GSS_GetMIC() ou GSS_Wrap() sur le contexte associé n'appliquera pas de protection cryptographique aux messages de données d'usager.

Les services GSS-API d'intégrité et d'authentification d'origine des données par message fournissent l'assurance à un

receveur que la protection a été appliquée à un message par l'homologue de l'appelant sur le contexte de sécurité, correspondant à l'entité désignée à l'initiation du contexte. Le service de confidentialité GSS-API par message donne l'assurance à un appelant envoyeur que le contenu du message est protégé contre l'accès par des entités autres que l'homologue désigné par le contexte.

Les primitives de service de protection GSS-API par message, comme l'implique le nom de catégorie, sont orientées vers un fonctionnement à une granularité d'unités de données de protocole. Elles effectuent les opérations cryptographiques sur les unités de données, transfèrent les informations de contrôle cryptographiques en jetons, et, dans le cas de GSS_Wrap(), encapsulent l'unité de données protégée. À ce titre, ces primitives ne sont pas orientée vers la protection efficace des données pour les protocoles par flux (par exemple, Telnet) si le chiffrement doit être appliqué sur la base de l'octet.

1.2.3 Détection de répétition et séquençage par message

Certains types de mécanisme sous-jacent offrent la prise en charge de la détection de répétition et/ou du séquençage des messages transférés sur les contextes qu'ils acceptent. Ces dispositifs de protection de choix facultatif sont distincts des dispositifs de détection de répétition et de séquençage appliqués à l'opération d'établissement de contexte elle-même ; la présence ou l'absence des dispositifs de répétition ou de séquençage de niveau contexte est totalement fonction des capacités du type de mécanisme sous-jacent, et n'est ni choisi ni omis comme option de l'appelant.

L'appelant qui initié un contexte fournit des fanions (replay_det_req_flag et sequence_req_flag) pour spécifier si l'utilisation des dispositifs de détection de répétition et de séquençage par message est désirée sur le contexte en cours d'établissement. La mise en œuvre GSS-API au système initiateur peut déterminer si ces dispositifs sont pris en charge (et si ils peuvent être choisis en option) comme fonction du mécanisme choisi, sans qu'il soit besoin d'une négociation bilatérale avec la cible. Lorsque ils sont activés, ces dispositifs fournissent des indicateurs au receveur comme résultat du traitement GSS-API des messages entrants, identifiant si ces messages ont été détectés comme dupliqués ou hors séquence. La détection de tels événements n'empêche pas un message suspect d'être fourni à un receveur ; l'action appropriée sur un message suspect est une affaire de politique de l'appelant.

La sémantique des services de détection de répétition et de séquençage appliquée aux messages reçus, tels que visibles à travers l'interface que fournit GSS-API à ses clients, est comme suit :

Lorsque replay_det_state est VRAI, les retours d'états majeurs possibles pour des messages bien formés et correctement signés sont comme suit :

1. GSS_S_COMPLETE, sans indication concurrente de GSS_S_DUPLICATE_TOKEN ou GSS_S_OLD_TOKEN, indique que le message était dans la fenêtre (d'espace de temps ou de séquence) permettant de détecter des événements de répétition, et que le message n'était pas une répétition d'un message précédemment traité dans cette fenêtre.
2. GSS_S_DUPLICATE_TOKEN indique que la valeur de vérification cryptographique sur le message reçu était correcte, mais que le message a été reconnu comme duplication d'un message précédemment traité. En plus d'identifier les jetons dupliqués générés par un homologue du contexte, cet état peut aussi être utilisé pour identifier des copies reflétées de jetons générés en local ; il est recommandé que les concepteurs de mécanisme incluent dans leurs protocoles des facilités pour détecter et faire rapport de tels jetons.
3. GSS_S_OLD_TOKEN indique que la valeur de vérification cryptographique sur le message reçu était correcte, mais que le message est trop vieux pour qu'on cherche s'il est dupliqué.

Lorsque l'état de séquence *(sequence_state)* est VRAI, les retours possibles d'état majeur pour des messages bien formés et correctement signés sont comme suit :

1. GSS_S_COMPLETE, sans indication concurrente de GSS_S_DUPLICATE_TOKEN, GSS_S_OLD_TOKEN, GSS_S_UNSEQ_TOKEN, ou GSS_S_GAP_TOKEN, indique que le message était dans la fenêtre (d'espace de temps ou de séquence) permettant de détecter les événements de répétition, que le message n'était pas une répétition d'un message précédemment traité au sein de cette fenêtre, et qu'aucun message d'une séquence précédente ne manque par rapport au dernier message reçu (si il en est) traité sur le contexte avec une valeur de vérification cryptographique correcte.
2. GSS_S_DUPLICATE_TOKEN indique que la valeur de vérification d'intégrité sur le message reçu était correcte, mais que le message a été reconnu comme duplication d'un message traité précédemment. En plus d'identifier les jetons dupliqués générés par un homologue du contexte, cet état peut aussi être utilisé pour identifier les copies reflétées de jetons générés en local ; il est recommandé que les concepteurs de mécanisme incluent dans leurs protocoles des facilités pour détecter et faire rapport de tels jetons.
3. GSS_S_OLD_TOKEN indique que la valeur de vérification d'intégrité sur le message reçu était correcte, mais que le jeton est trop vieux pour qu'on vérifie si c'est un dupliqué.
4. GSS_S_UNSEQ_TOKEN indique que la valeur de vérification cryptographique sur le message reçu était correcte, mais qu'il est plus tôt dans un flux séquentiel qu'un message déjà traité sur le contexte. [Note : les mécanismes peuvent être construits de façon à fournir une forme plus stricte de service de séquençage, ne livrant des messages particuliers aux

receveurs qu'après que tous les messages précédents dans un flux ordonné ont été livrés. Ce type de support est incompatible avec le paradigme GSS-API dans lequel les receveurs reçoivent tous les messages, dans l'ordre ou non, et les fournissent (un à la fois, sans mise en mémoire tampon intra GSS-API) aux sous-programmes GSS-API pour validation. Les facilités de GSS-API fournissent des fonctions de soutien qui aident les clients à réaliser une stricte intégrité du flux de messages d'une manière efficace en conjonction avec des dispositions de séquençage dans les protocoles de communication, mais GSS-API n'offre pas par lui-même ce niveau de service d'intégrité de flux de messages.]

5. GSS_S_GAP_TOKEN indique que la valeur de vérification cryptographique sur le message reçu était correcte, mais que un ou plusieurs messages avec des numéros de séquence antérieurs n'ont pas été traités avec succès par rapport au dernier message reçu (si il en est) traité sur le contexte avec une valeur de vérification cryptographique correcte.

Comme les dispositifs d'intégrité de flux de messages (en particulier de séquençage) peuvent interférer avec certains paradigmes de communication prévus par les applications, et comme la prise en charge de tels dispositifs sera vraisemblablement gros consommateur de ressources, il est fortement recommandé que les types de mécanismes qui prennent en charge ces dispositifs leur permettent d'être activés de façon sélective à la demande de l'initiateur lorsque un contexte est établi. Des indicateurs correspondants sont fournis à l'initiateur de contexte et à la cible (replay_det_state et sequence_state) pour signifier si ces dispositifs sont actifs sur un certain contexte.

Un exemple de mech_type qui prend en charge la détection par message de la répétition pourrait (lorsque replay_det_state est VRAI) mettre en œuvre le dispositif comme suit : le mécanisme sous-jacent insèrerait des horodatages dans les éléments de données produits par GSS_GetMIC() et GSS_Wrap(), et entretiendrait (dans une fenêtre temporelle limitée) une antémémoire (qualifiée par la paire origine-receveur) qui identifierait les éléments de données reçus traités par GSS_VerifyMIC() et GSS_Unwrap(). Lorsque ce dispositif est actif, des retours d'état d'exception (GSS_S_DUPLICATE_TOKEN, GSS_S_OLD_TOKEN) vont être fournis lorsque GSS_VerifyMIC() ou GSS_Unwrap() est présenté avec un message qui est soit un dupliqué détecté d'un message antérieur, soit est trop vieux pour être validé contre une antémémoire des messages reçus récemment.

1.2.4 Qualité de protection

Certains types de mécanisme fournissent à leurs utilisateurs un contrôle d'une granularité fine sur les moyens utilisés pour assurer la protection par message, permettant aux appelants de faire un compromis dynamique entre la surcharge du traitement de la sécurité et les exigences de protection de certains messages. Un paramètre de qualité de protection par message (analogue à la qualité de service, ou QS) choisit parmi différentes options de QOP prises en charge par ce mécanisme. À l'établissement de contexte pour un type de mécanisme multi QOP, les données de niveau contexte fournissent les données de prérequis pour une gamme de qualités de protection.

On s'attend à ce qu'une majorité d'appelants ne souhaitent pas exercer un contrôle explicite de QOP spécifique du mécanisme et vont donc demander le choix de la QOP par défaut. Les définitions, et les choix parmi les valeurs de QOP qui ne sont pas celle par défaut sont spécifiques du mécanisme, et aucune séquence ordonnée de valeurs de QOP ne peut être supposée équivalente entre des mécanismes différents. Pour que l'utilisation de valeurs de QOP autres que par défaut ait une signification, il faut que les appelants soient familiarisés avec les définitions de QOP du ou des mécanismes sous-jacents, et c'est donc une construction non portable. La valeur de l'état majeur GSS_S_BAD_QOP est définie afin d'indiquer qu'une valeur de QOP fournie n'est pas prise en charge pour un contexte de sécurité, très vraisemblablement parce que cette valeur n'est pas reconnue par le mécanisme sous-jacent.

Dans l'intérêts de l'interopérabilité, les mécanismes qui permettent la prise en charge facultative de valeurs de QOP particulières devraient satisfaire à une des conditions suivantes :

(i) toutes les mises en œuvre du mécanisme doivent être capables de traiter les messages protégés en utilisant toute valeur de QOP, sans considérer si elles peuvent appliquer la protection correspondant à cette QOP, ou

(ii) l'ensemble des valeurs de QOP mutuellement prises en charge en réception doit être déterminé durant l'établissement de contexte, et les messages peuvent être protégés par l'un ou l'autre homologue en utilisant seulement les valeurs de QOP de cet ensemble mutuellement pris en charge.

Note : (i) est juste un cas particulier de (ii), où les mises en œuvre doivent prendre en charge toutes les valeurs de QOP à réception.

1.2.5 Prise en charge de l'anonymat

Dans certaines situations ou environnements, une application peut souhaiter authentifier un homologue et/ou protéger les communications en utilisant les services GSS-API par message sans révéler sa propre identité. Par exemple, considérons une application qui fournit un accès en lecture à une base de données de recherche, et qui permet à n'importe qui de poser des questions. Un client d'un tel service peut souhaiter authentifier le service, pour établir la confiance dans les

informations qu'elle en reçoit, mais peut ne pas souhaiter dévoiler son identité au service pour des raisons de confidentialité.

Dans l'utilisation ordinaire de GSS-API, l'identité d'un initiateur de contexte est rendue disponible à l'accepteur de contexte au titre du processus d'établissement de contexte. Pour assurer la prise en charge de l'anonymat est fournie une facilité (entrée de anon_req_flag dans GSS_Init_sec_context()) par laquelle un initiateur de contextes peut demander que son identité ne soit pas fournie à l'accepteur de contexte. Les mécanismes ne sont pas obligés d'honorer cette demande, mais un appelant sera informé (via l'indicateur anon_state retourné par GSS_Init_sec_context()) si la demande est ou non satisfaite. Noter que l'authentification comme principal anonyme n'implique pas nécessairement que les accréditifs ne soient pas exigés afin d'établir un contexte.

Le paragraphe 4.5 de ce document définit la valeur de l'identifiant d'objet utilisée pour identifier un principal anonyme.

Quatre combinaisons possibles de anon_state et mutual_state existent, avec le résultat suivant :
anon_state == FAUX, mutual_state == FAUX : initiateur authentifié auprès de la cible.
anon_state == FAUX, mutual_state == VRAI : initiateur authentifié auprès de la cible, cible authentifiée auprès de l'initiateur.
anon_state == VRAI, mutual_state == FAUX : initiateur authentifié comme principal anonyme auprès de la cible.
anon_state == VRAI, mutual_state == VRAI : initiateur authentifié comme principal anonyme auprès de la cible, cible authentifiée auprès de l'initiateur.

1.2.6 Initialisation

Aucun appel d'initialisation (c'est-à-dire, d'appel qui doive être invoqué avant d'autres facilités dans l'interface) n'est défini dans GSS-API. Ce fait implique que les mises en œuvre de GSS-API doivent s'initialiser elles-mêmes.

1.2.7 Protection par message durant l'établissement de contexte

Une facilité est définie dans GSS-V2 pour permettre la protection et la mise en mémoire tampon des messages de données pour transfert ultérieur lorsque l'établissement d'un contexte de sécurité est dans l'état GSS_S_CONTINUE_NEEDED, pour être utilisés dans les cas où le côté appelant possède déjà la clé de session nécessaire pour permettre ce traitement. Précisément, un nouvel état booléen, appelé prot_ready_state *(état de protection prêt)*, est ajouté à l'ensemble des informations retournées par GSS_Init_sec_context(), GSS_Accept_sec_context(), et GSS_Inquire_context().

Pour les appels d'établissement de contexte, cet état booléen est valide et interprétable lorsque l'état majeur associé est soit GSS_S_CONTINUE_NEEDED, soit GSS_S_COMPLETE. Les appelants de GSS-API (initiateurs et accepteurs) peuvent supposer que la protection par message (via GSS_Wrap(), GSS_Unwrap(), GSS_GetMIC() et GSS_VerifyMIC()) est disponible et prête à l'usage si : prot_ready_state == VRAI, ou major_status == GSS_S_COMPLETE, bien que l'authentification mutuelle (si elle est demandée) ne puisse être garantie jusqu'à ce que GSS_S_COMPLETE soit retourné. Les appelants qui font usage des services de protection par message avant l'état GSS_S_COMPLETE devraient être conscients de la possibilité qu'une étape suivante de l'établissement de contexte puisse échouer, et que certaines données de contexte (par exemple, le mech_type) qui sont retournées pour les appels suivants peuvent changer.

Cette approche réalise une pleine rétro compatibilité transparente pour les appelants GSS-API V1 qui n'ont même pas besoin de connaître l'existence de prot_ready_state, et qui vont avoir le comportement attendu de la part de GSS_S_COMPLETE, mais qui ne seront pas capables d'utiliser la protection par message avant le retour de GSS_S_COMPLETE.

Il n'est pas exigé que le mécanisme GSS-v2 retourne toujours VRAI pour prot_ready_state avant l'achèvement de l'établissement de contexte (bien sûr, certains mécanismes ne vont pas faire évoluer les clé de protection de message utilisables, en particulier chez l'accepteur de contexte, avant l'achèvement de l'établissement de contexte). On s'attend, bien que ce ne soit pas exigé, à ce que le mécanisme GSS-v2 retourne VRAI pour prot_ready_state à l'achèvement de l'établissement de contexte si il prend bien en charge la protection par message (cependant, les applications GSS-v2 ne devraient pas supposer que VRAI pour prot_ready_state sera toujours retourné avec l'état majeur GSS_S_COMPLETE, car les mises en œuvre GSS-v2 peuvent continuer de prendre en charge le code de mécanisme GSS-V1, qui ne va jamais retourner VRAI pour prot_ready_state).

Lorsque prot_ready_state est retourné avec la valeur VRAI, les mécanismes devront aussi établir les fanions d'indicateur de service de contexte (deleg_state, mutual_state, replay_det_state, sequence_state, anon_state, trans_state, conf_avail, integ_avail) qui représentent les facilités confirmées, pour l'instant, comme étant disponibles sur le contexte en cours d'établissement. Dans les situations où prot_ready_state est retourné avant GSS_S_COMPLETE, il est possible que des facilités supplémentaires puissent être confirmées et indiquées ensuite lorsque GSS_S_COMPLETE est retourné.

1.2.8 Robustesse de mise en œuvre

Ce paragraphe fait des recommandations sur les aspects du comportement des mises en œuvre de GSS-API qui concernent la robustesse globale.

L'invocation des appels GSS-API ne provoque aucun effet collatéral non documenté visible au niveau de GSS-API.

Si un jeton est présenté au traitement sur un contexte de sécurité GSS-API et si le jeton génère une erreur fatale pendant le traitement ou est par ailleurs déterminé invalide pour ce contexte, l'état du contexte ne devrait pas être interrompu pour les besoins du traitement des jetons valides suivants.

Certaines conditions locales d'une mise en œuvre de GSS-API (par exemple, mémoire indisponible) peuvent empêcher, temporairement ou de façon permanente, la réussite du traitement des jetons sur un contexte de sécurité GSS-API, générant normalement des retours d'état majeur GSS_S_FAILURE avec des états mineurs de signification locale. Pour un fonctionnement robuste dans de telles conditions, on formule les recommandations suivantes :

- les échecs d'appel devraient libérer toutes les ressources de mémoire qui lui étaient allouées, de sorte que les appelants puissent réessayer sans causer d'autres pertes de ressources ;
- l'échec d'un appel individuel sur un contexte établi ne devrait pas empêcher la réussite d'appels ultérieurs sur le même contexte ;
- chaque fois que possible, les appels GSS_Delete_sec_context() devraient pouvoir être traités avec succès même si d'autres appels ne peuvent pas réussir, permettant par là la libération des ressources relatives au contexte.

Une défaillance de GSS_GetMIC() ou GSS_Wrap() due à une tentative d'utilisation d'une QOP non prise en charge ne va pas interférer avec la validité du contexte, et une telle défaillance ne devra pas avoir d'impact sur la capacité de l'application à invoquer ultérieurement GSS_GetMIC() ou GSS_Wrap() en utilisant une QOP acceptée. Toutes les informations d'état concernant le séquençage des messages sortants devront être inchangées par un échec d'appel de GSS_GetMIC() ou GSS_Wrap().

1.2.9 Délégation

GSS-API permet que la délégation soit contrôlée par l'application initiatrice via un paramètre booléen à GSS_Init_sec_context(), le sous-programme qui établit un contexte de sécurité. Certains mécanismes ne prennent pas en charge la délégation, et pour de tels mécanismes les tentatives d'une application pour activer la délégation sont ignorées.

L'accepteur d'un contexte de sécurité pour lequel l'initiateur active la délégation va recevoir (via le paramètre delegated_cred_handle de GSS_Accept_sec_context()) un lien d'accréditif qui contient l'identité déléguée, et ce lien d'accréditif peut être utilisé pour initier les contextes de sécurité GSS-API suivants comme un agent ou délégué de l'initiateur. Si l'identité originelle de l'initiateur est "A" et si l'identité du délégué est "B", alors, selon le mécanisme sous-jacent, l'identité incorporée par l'accréditif délégué peut être soit "A" soit "B agissant pour A".

Pour de nombreux mécanismes qui prennent en charge la délégation, un simple booléen ne fournit pas assez de contrôle. Des exemples des aspects supplémentaires de contrôle de délégation que pourrait fournir un mécanisme à une application sont la durée de délégation, les adresses réseau à partir desquelles la délégation est valide, et les contraintes sur les tâches qui peuvent être effectuées par le délégué. De tels contrôles sont actuellement en dehors du domaine d'application de GSS-API. Les mises en œuvre de GSS-API qui acceptent les mécanismes offrant des contrôles supplémentaires devraient fournir des sous-programmes d'extension permettant d'exécuter des contrôles (peut-être en modifiant l'accréditif GSS-API de l'initiateur avant son utilisation dans l'établissement d'un contexte). Cependant, le simple contrôle de délégation fourni par GSS-API devrait toujours être capable d'outrepasser les contrôles de délégation spécifiques des autres mécanismes ; si l'application indique à GSS_Init_sec_context() que la délégation n'est pas désirée, la mise en œuvre ne doit alors pas permettre à la délégation de se produire. Ceci est une exception à la règle générale qu'un mécanisme peut activer des services même si ils ne sont pas demandés ; la délégation ne peut être fournie qu'à la demande explicite de l'application.

1.2.10 Transfert de contexte inter processus

GSS-API V2 fournit des sous-programmes (GSS_Export_sec_context() et GSS_Import_sec_context()) qui permettent qu'un contexte de sécurité soit transféré entre des processus sur une seule machine. L'utilisation la plus courante d'un tel dispositif est un concept client-serveur où le serveur est mis en œuvre comme un seul processus qui accepte les contextes de sécurité entrants, qui lance alors les processus fils pour traiter les données sur ces contextes. Dans un tel concept, le processus fils doit avoir accès à la structure des données du contexte de sécurité créée au sein du parent par son invocation de GSS_Accept_sec_context() afin qu'il puisse utiliser les services de protection par message et supprimer le contexte de sécurité lorsque se termine la session de communication.

Comme la structure de données du contexte de sécurité est censée contenir les informations de séquençage, il n'est pas possible en général de partager un contexte entre des processus. Donc, GSS-API fournit une invocation (GSS_Export_sec_context()) que le processus qui détient actuellement le contexte peut invoquer pour déclarer qu'il n'a pas l'intention d'utiliser ultérieurement le contexte, et de créer un jeton inter procès contenant les informations nécessaires au processus adopteur pour réussir à importer le contexte. Après l'achèvement réussi de cette invocation, le contexte de sécurité originel est rendu inaccessible au processus appelant par GSS-API, et tout lien de contexte se référant à ce contexte devient invalide. Le processus générateur transfère le jeton inter procès au processus adopteur, qui le passe à GSS_Import_sec_context(), et un lien de contexte frais est créé de telles sorte qu'il soit fonctionnellement identique au contexte originel.

Le jeton inter procès peut contenir des données sensibles provenant du contexte de sécurité originel (incluant des clés cryptographiques). Les applications qui utilisent des jetons inter procès pour transférer des contextes de sécurité doivent prendre les mesures appropriées pour protéger ces jetons dans le transit. Les mises en œuvre ne sont pas obligées d'accepter le transfert inter procès de contextes de sécurité. La capacité à transférer un contexte de sécurité est indiquée lorsque le contexte est créé, par GSS_Init_sec_context() ou GSS_Accept_sec_context() indiquant une valeur VRAI de retour trans_state.

2. Descriptions d'interface

Cette section décrit l'interface de service de GSS-API en divisant l'ensemble des appels offerts en quatre groupes. Les appels de gestion d'accréditifs se rapportent à l'acquisition et la libération des accréditifs par les principaux. Les appels de niveau contexte se rapportent à la gestion des contextes de sécurité entre les principaux. Les appels par message se rapportent à la protection des messages individuels sur des contextes de sécurité établis. Les appels de soutien fournissent des fonctions auxiliaires utiles aux appelants GSS-API. Le Tableau 2 regroupe et résume les quatre groupes d'appels.

Tableau 2 : Invocations de GSS-API

Gestion des accréditifs	
GSS_Acquire_cred	acquérir des accréditifs à utiliser
GSS_Release_cred	libérer les accréditifs après utilisation
GSS_Inquire_cred	afficher les informations sur les accréditifs
GSS_Add_cred	construire les accréditifs de façon incrémentaire
GSS_Inquire_cred_by_mech	afficher les informations d'accréditif par mécanisme
Appels de niveau contexte	
GSS_Init_sec_context	initier le contexte de sécurité sortant
GSS_Accept_sec_context	accepter le contexte de sécurité entrant
GSS_Delete_sec_context	purger le contexte quand il n'est plus utile
GSS_Process_context_token	traiter le jeton de contrôle reçu sur le contexte
GSS_Context_time	indiquer la durée de validité restante sur le contexte
GSS_Inquire_context	afficher les informations sur le contexte
GSS_Wrap_size_limit	déterminer la limite de taille du jeton GSS_Wrap
GSS_Export_sec_context	transférer le contexte à un autre processus
GSS_Import_sec_context	importer le contexte transféré
Appels par message	
GSS_GetMIC	appliquer la vérification d'intégrité, reçue comme jeton séparé du message
GSS_VerifyMIC	valider le jeton de vérification d'intégrité avec le message
GSS_Wrap	signe, chiffre facultativement, encapsule
GSS_Unwrap	désencapsule, déchiffre si nécessaire, valide la vérification d'intégrité
Appels de soutien	
GSS_Display_status	traduit les codes d'état en forme imprimable
GSS_Indicate_mechs	indique les types de mécanisme acceptés sur le système local
GSS_Compare_name	compare l'égalité de deux noms
GSS_Display_name	traduit le nom en forme imprimable
GSS_Import_name	convertit le nom imprimable en forme normalisée
GSS_Release_name	mémorisation libre de nom en forme normalisée
GSS_Release_buffer	mémorisation libre d'objet général alloué par GSS
GSS_Release_OID_set	mémorisation libre d'objet d'ensemble d'OID

GSS_Create_empty_OID_set	créer un ensemble d'OID vide
GSS_Add_OID_set_member	ajouter des membres à l'ensemble d'OID
GSS_Test_OID_set_member	vérifier si l'OID est membre de l'ensemble d'OID
GSS_Inquire_names_for_mech	indique les types de noms acceptés par le mécanisme
GSS_Inquire_mechs_for_name	indique les mécanismes qui prennent en charge le type de noms
GSS_Canonicalize_name	traduit le nom en forme par mécanisme
GSS_Export_name	externalise le nom par mécanisme
GSS_Duplicate_name	objet nom dupliqué

2.1 Appels de gestion d'accréditifs

Ces appels GSS-API fournissent des fonctions qui se rapportent à la gestion des accréditifs. Leur caractérisation par rapport au fait qu'ils peuvent ou non bloquer les échanges en cours avec d'autres entités réseau (par exemple, les répertoires ou les serveurs d'authentification) dépend en partie de questions spécifiques du système d'exploitation (extra GSS-API) et n'est donc pas spécifiée dans ce document.

L'invocation GSS_Acquire_cred() est définie dans GSS-API en soutien de la portabilité d'application, avec une orientation particulière vers le soutien des applications de serveur portable. On reconnaît que (pour certains systèmes et mécanismes) les accréditifs pour les utilisateurs interactifs peuvent être gérés différemment des accréditifs pour les processus de serveur ; dans de tels environnements, il est de la responsabilité de la mise en œuvre de GSS-API de distinguer ces cas et les procédures pour faire cette distinction sont une affaire locale. L'invocation GSS_Release_cred() donne à l'appelant le moyen d'indiquer à GSS-API que l'utilisation d'une structure d'accréditifs n'est plus exigée. L'invocation de GSS_Inquire_cred() permet aux appelants de déterminer les informations sur une structure d'accréditifs. L'invocation de GSS_Add_cred() permet aux appelants d'ajouter des éléments à une structure d'accréditifs existante, permettant une construction itérative d'un accréditif multi mécanisme. L'invocation de GSS_Inquire_cred_by_mech() permet aux appelants d'extraire des informations par mécanisme qui décrivent une structure d'accréditif.

2.1.1 Invocation de GSS_Acquire_cred

Entrées :

- NOM ENTIER desired_name, -- NUL demande une valeur par défaut déterminée en local
- ENTIER lifetime_req, -- en secondes ; 0 demande la valeur par défaut
- ENSEMBLE D'IDENTIFIANT D'OBJET desired_mechs, -- NUL demande une valeur par défaut choisie par le système
- ENTIER cred_usage, -- 0 = INITIER ET ACCEPTER, 1 = INITIER SEULEMENT, 2 = ACCEPTER SEULEMENT

Sorties :

- ENTIER major_status,
- ENTIER minor_status,
- LIEN D'ACCRÉDITIF output_cred_handle, -- si un non NUL est retourné, l'appelant doit libérer avec GSS_Release_cred()
- ENSEMBLE D'IDENTIFIANT D'OBJET actual_mechs, -- si un non NUL est retourné, l'appelant doit libérer avec GSS_Release_oid_set()
- ENTIER lifetime_rec, -- en secondes, ou valeur réservée pour INDÉFINI

Codes d'états majeurs en retour :

- GSS_S_COMPLETE indique que les accréditifs demandés ont été bien établis, pour la durée indiquée dans lifetime_rec, convenables pour l'usage demandé dans cred_usage, pour l'ensemble de types de mécanismes indiqué dans actual_mechs, et que ces accréditifs peuvent être référencés pour les usages ultérieurs avec le lien retourné dans output_cred_handle.
- GSS_S_BAD_MECH indique qu'un type de mécanisme non accepté par le type de mise en œuvre GSS-API a été demandé, causant l'échec de l'opération d'établissement d'accréditif.
- GSS_S_BAD_NAMETYPE indique que le nom désiré fourni n'est pas interprétable ou est d'un type non accepté par le ou les mécanismes GSS-API sous-jacents applicables, de sorte qu'aucun accréditif n'a pu être établi pour le desired_name qui l'accompagne.
- GSS_S_BAD_NAME indique que le nom désiré fourni n'est pas cohérent en termes d'informations de spécificateur de type incorporé en interne, de sorte qu'aucun accréditif n'a pu être établi pour le desired_name qui l'accompagne.
- GSS_S_CREDENTIALS_EXPIRED indique que les éléments d'accréditif sous-jacents correspondant au nom désiré demandé sont arrivés à expiration, de sorte que les accréditifs demandés n'ont pas pu être établis.
- GSS_S_NO_CRED indique qu'on n'a pas pu accéder aux éléments d'accréditifs correspondant au nom et à l'usage désiré demandés, de sorte que les accréditifs demandés n'ont pas pu être établis. En particulier, cet état devrait être

retourné lorsque des conditions temporaires fixées par l'usager empêchent la réussite de l'établissement des accréditifs et lors d'un manque d'autorisation d'établir et utiliser les accréditifs associés à l'identité désignée dans l'argument d'entrée desired_name.

- GSS_S_FAILURE indique que l'établissement d'accréditifs a échoué pour des raisons non spécifiées au niveau GSS-API.

GSS_Acquire_cred() est utilisé pour acquérir des accréditifs afin qu'un principal puisse (en fonction du paramètre d'entrée cred_usage) initier et/ou accepter des contextes de sécurité sous l'identité représentée par l'argument d'entrée desired_name. Sur achèvement réussi, le résultat output_cred_handle retourné fournit un lien pour les références ultérieures aux accréditifs acquis. Normalement, un client d'un seul usager qui le traite en demandant que le comportement d'accréditif par défaut soit appliqué pour les besoins de l'établissement de contexte n'aura pas besoin d'invoquer cet appel.

Un appelant peut fournir la valeur NUL (GSS_C_NO_NAME) pour le nom désiré, ce qui sera interprété comme la demande d'un lien d'accréditif qui va invoquer le comportement par défaut lorsque il sera passé à GSS_Init_sec_context(), si cred_usage est GSS_C_INITIATE ou GSS_C_BOTH, ou GSS_Accept_sec_context(), si cred_usage est GSS_C_ACCEPT ou GSS_C_BOTH. Il est possible que plusieurs accréditifs préétablis existent pour la même identité de principal (par exemple, par suite de plusieurs sessions de connexion d'usager) lorsque GSS_Acquire_cred() est invoqué ; les moyens utilisés dans de tels cas pour choisir un accréditif spécifique sont une affaire locale. L'argument d'entrée lifetime_req à GSS_Acquire_cred() peut fournir des informations utiles pour les mises en œuvre GSS-API locales à employer pour faire cette précision d'une manière qui satisfasse au mieux l'intention de l'appelant.

Ce sous-programme est supposé être utilisé principalement par les accepteurs de contexte, car les mises en œuvre vont probablement fournir des façons spécifiques du mécanisme pour obtenir les accréditifs d'initiateur GSS-API du processus de connexion du système. Certaines mises en œuvre peuvent donc ne pas prendre en charge l'acquisition des accréditifs GSS_C_INITIATE ou GSS_C_BOTH via GSS_Acquire_cred() pour tout nom autre que GSS_C_NO_NAME, ou un nom résultant de l'application de GSS_Inquire_context() à un contexte actif, ou un nom résultant de l'application de GSS_Inquire_cred() à un lien d'accréditif correspondant à un comportement par défaut. Il est important de reconnaître que le nom explicite qui est donné par la résolution d'une référence par défaut peut changer au fil du temps, par exemple, par suite des opérations locales de gestion d'élément d'accréditif en-dehors de GSS-API ; une fois résolue, la valeur d'un tel nom explicite va cependant rester constante.

Le résultat lifetime_rec indique la durée pendant laquelle les accréditifs acquis vont être valides, comme un décalage par rapport au présent. Un mécanisme peut retourner une valeur réservée qui indique INDÉFINI si aucune contrainte n'est imposée à la durée de vie de l'accréditif. Un appelant de GSS_Acquire_cred() peut demander une durée pour laquelle les accréditifs acquis seront valides (argument lifetime_req) commençant au présent, ou peut demander des accréditifs avec un intervalle de validité par défaut. (Les demandes d'accréditifs postdatés ne sont pas acceptées au sein de GSS-API.) Certains mécanismes et certaines mises en œuvre peuvent lier des spécificateurs de période de validité d'accréditif à un instant antérieur à l'invocation de GSS_Acquire_cred() (par exemple, en conjonction avec les procédures de connexion de l'usager). Il en résulte que les appelants qui demandent des valeurs qui ne sont pas par défaut pour lifetime_req doivent reconnaître que de telles demandes ne peuvent pas toujours être honorées et doivent être prêts à s'accommoder de l'utilisation d'accréditifs retournés avec es durées de vie différentes de ce qui est indiqué dans lifetime_rec.

L'appelant de GSS_Acquire_cred() peut explicitement spécifier un ensemble de mech_types qui seront accommodés dans les accréditifs retournés (argument desired_mechs) ou peut demander des accréditifs pour un ensemble par défaut de mech_types défini par le système. Le choix de l'ensemble par défaut spécifié par le système est recommandé dans l'intérêt de la portabilité de l'application. La valeur de retour de actual_mechs peut être interrogée par l'appelant pour déterminer l'ensemble des mécanismes avec lesquels les accréditifs retournés peuvent être utilisés.

2.1.2 Invocation de GSS_Release_cred

Entrée :

- LIEN D'ACCRÉDITIF cred_handle - si GSS_C_NO_CREDENTIAL est spécifié, l'appel va bien se terminer, mais n'aura pas d'effet ; aucun élément d'accréditif ne sera libéré.

Sorties :

- ENTIER major_status,
- ENTIER minor_status.

Codes d'état majeur en retour :

- GSS_S_COMPLETE indique que les accréditifs référencés par l'entrée cred_handle ont été libérés pour les besoins d'accès ultérieurs par l'appelant. L'effet sur les autres processus qui peuvent avoir un accès autorisé partagé à de tels accréditifs est une affaire locale.

- o GSS_S_NO_CRED indique qu'aucune opération de libération n'a été effectuée, soit parce que l'entrée cred_handle était invalide, soit parce que l'appelant n'a pas l'autorisation d'accéder aux accréditifs référencés.
- o GSS_S_FAILURE indique que l'opération de libération a échoué pour des raisons non spécifiées au niveau GSS-API.

Il n'est plus exigé de fournir à un appelant un moyen de demander explicitement que les accréditifs soient libérés lorsque leur utilisation n'est plus exigée. Noter que il est aussi probable qu'existent des fonctions de gestion d'accréditifs spécifiques du système, par exemple pour s'assurer que les accréditifs partagés entre les processus sont correctement supprimés lorsque tous les processus affectés sont terminés, même si aucune demande explicite de libération n'est produite par ces processus. Compte tenu du fait que plusieurs appelants peuvent obtenir un accès autorisé aux mêmes accréditifs, l'invocation de GSS_Release_cred() ne peut pas être supposée supprimer un ensemble particulier d'accréditifs sur la base de tout un système.

2.1.3 Invocation de GSS_Inquire_cred

Entrée :

- o LIEN D'ACCRÉDITIF cred_handle -- si GSS_C_NO_CREDENTIAL est spécifié, les accréditifs d'initiateur par défaut sont interrogés.

Sorties :

- o ENTIER major_status ,
- o ENTIER minor_status,
- o NOM INTERNE cred_name, -- l'appelant doit libérer avec GSS_Release_name().
- o ENTIER lifetime_rec -- en secondes, ou valeur réservée pour INDÉFINI.
- o ENTIER cred_usage, -- 0=INITIE-ET-ACCEPTE, 1=INITIE-SEUL, 2=ACCEPTE-SEUL
- o ENSEMBLE D'IDENTIFIANTS D'OBJET mech_set -- l'appelant doit libérer avec GSS_Release_oid_set()

Codes d'état majeur en retour :

- o GSS_S_COMPLETE indique que les accréditifs référencés par l'argument d'entrée cred_handle étaient valides, et que les valeurs de sortie cred_name, lifetime_rec, et cred_usage représentent, respectivement, le nom de principal associé, la durée de vie restante, les modes d'usage convenables, et les types de mécanismes pris en charge, des accréditifs.
- o GSS_S_NO_CRED indique qu'aucune information n'a pu être retournée sur les accréditifs référencés, soit parce que le cred_handle d'entrée était invalide, soit parce que l'appelant n'a pas l'autorisation d'accès aux accréditifs référencés.
- o GSS_S_DEFECTIVE_CREDENTIAL indique que les accréditifs référencés sont invalides.
- o GSS_S_CREDENTIALS_EXPIRED indique que les accréditifs référencés sont périmés.
- o GSS_S_FAILURE indique que l'opération a échoué pour des raisons non spécifiées au niveau GSS-API.

L'appel GSS_Inquire_cred() est défini principalement pour être utilisé par les appelants qui demandent l'utilisation du comportement d'accréditif par défaut plutôt que d'acquérir explicitement des accréditifs avec GSS_Acquire_cred(). Il permet à l'appelant de déterminer le nom du principal associé, la période de validité restante, la possibilité d'utilisation pour l'initiation et/ou l'acceptation du contexte de sécurité, et les mécanismes pris en charge, d'une structure d'accréditifs.

Pour un accréditif multi mécanismes, le spécificateur de la "durée de vie" retournée indique la plus courte durée de vie de tous les éléments des mécanismes de l'accréditif (pour les besoins de l'initiation ou de l'acceptation de contexte).

GSS_Inquire_cred() devrait indiquer INITIE-ET-ACCEPTE pour "cred_usage" si les deux conditions suivantes tiennent :

(1) il existe dans l'accréditif un élément qui permet l'initiation de contexte en utilisant un mécanisme,

(2) il existe dans l'accréditif un élément qui permet l'acceptation de contexte utilisant un mécanisme (permis, mais pas nécessairement un des mêmes mécanismes qualifiés pour (1)).

Si la condition (1) tient mais pas la condition (2), GSS_Inquire_cred() devrait indiquer INITIE-SEUL pour "cred_usage". Si la condition (2) tient mais pas la condition (1), GSS_Inquire_cred() devrait indiquer ACCEPTE-SEUL pour "cred_usage".

Les appelants qui demandent une précision plus grande sur les combinaisons disponibles de durée de vie, modes d'usage, et mécanismes devraient invoquer le sous-programme GSS_Inquire_cred_by_mech(), en passant sur ce sous-programme l'OID de mécanisme retourné par GSS_Inquire_cred().

2.1.4 Invocation de GSS_Add_cred

Entrées :

- o LIEN D'ACCRÉDITIF input_cred_handle -- lien à la structure d'accréditif créée avec l'invocation antérieure de GSS_Acquire_cred() ou GSS_Add_cred() ; voir le texte pour la définition du comportement lorsque GSS_C_NO_CREDENTIAL est

fourni.

- o NOM INTERNE desired_name
- o ENTIER initiator_time_req -- en secondes ; 0 demande la valeur par défaut.
- o ENTIER acceptor_time_req -- en secondes ; 0 demande la valeur par défaut.
- o IDENTIFIANT D'OBJET desired_mech
- o ENTIER cred_usage -- 0=INITIE-ET-ACCEPTE, 1=INITIE-SEUL, 2=ACCEPTE-SEUL

Sorties :

- o ENTIER major_status,
- o ENTIER minor_status,
- o LIEN D'ACCRÉDITIF output_cred_handle, -- NUL pour demander que les éléments d'accréditif soient ajoutés "à la place" de la structure d'accréditif identifiée par input_cred_handle, pointeur non NUL pour demander qu'une nouvelle structure et lien d'accréditif soient créés. Si le lien d'accréditif est retourné, l'appelant doit libérer avec GSS_Release_cred().
- o ENSEMBLE D'IDENTIFIANTS D'OBJET actual_mechs, -- Ensemble complet des mécanismes pris en charge par l'accréditif résultant. Si il est retourné, l'appelant doit libérer avec GSS_Release_oid_set()
- o ENTIER initiator_time_rec -- en secondes, ou valeur réservée pour INDÉFINI.
- o ENTIER acceptor_time_rec -- en secondes, ou valeur réservée pour INDÉFINI.

Codes d'état majeur en retour :

- o GSS_S_COMPLETE indique que les accréditifs référencés par l'argument input_cred_handle étaient valides, et que l'accréditif résultant de GSS_Add_cred() est valide pour les durées indiquées dans initiator_time_rec et acceptor_time_rec, convenables pour l'usage demandé dans cred_usage, et pour le mécanismes indiqué dans actual_mechs.
- o GSS_S_DUPLICATE_ELEMENT indique que l'entrée desired_mech spécifiait un mécanisme pour lequel l'accréditif référencé contient déjà un élément d'accréditif avec un cred_usage et des spécificateurs de durée de validité qui se chevauchent.
- o GSS_S_BAD_MECH indique que l'entrée desired_mech spécifiait un mécanisme non pris en charge par la mise en œuvre GSS-API, causant l'échec de l'opération GSS_Add_cred().
- o GSS_S_BAD_NAMETYPE indique que le desired_name fourni n'est pas interprétable ou d'un type non pris en charge par le ou les mécanismes GSS-API sous-jacents applicables, de sorte que l'opération GSS_Add_cred() n'a pas pu être effectuée pour ce nom.
- o GSS_S_BAD_NAME indique que le desired_name fourni n'est pas cohérent en termes d'informations de spécificateur de type incorporées en interne, de sorte que l'opération GSS_Add_cred() n'a pas pu être effectuée pour ce nom.
- o GSS_S_NO_CRED indique que le input_cred_handle fait référence à des accréditifs invalides ou inaccessibles. En particulier, cet état devrait être retourné lorsque des conditions temporaires fixées par l'usager empêchent la réussite de l'établissement de l'accréditif ou en l'absence d'autorisation d'établir ou utiliser les accréditifs représentant l'identité demandée.
- o GSS_S_CREDENTIALS_EXPIRED indique que les éléments d'accréditif référencés sont périmés, de sorte que l'opération GSS_Add_cred() n'a pas pu être effectuée.
- o GSS_S_FAILURE indique que l'opération a échoué pour des raisons non spécifiées au niveau GSS-API.

GSS_Add_cred() permet aux appelants de construire itérativement des accréditifs par l'ajout d'éléments d'accréditif en opérations successives, correspondant aux différents mécanismes. Cela est particulièrement précieux dans les environnements multi mécanismes, car les valeurs de major_status et minor_status retournées sur chaque itération sont individuellement visibles et peuvent donc être interprétées sans ambiguïté mécanisme par mécanisme. Un élément d'accréditif est identifié par le nom du principal auquel il se réfère. Les mises en œuvre GSS-API doivent imposer aux appelants de ce sous-programme une politique de contrôle d'accès locale pour empêcher les appelants non autorisés d'acquérir des éléments d'accréditif auxquels ils n'ont pas droit. Ce sous programme n'est pas destiné à fournir une fonction de "connexion au réseau", car une telle fonction impliquerait la création de nouvelles données d'authentification spécifiques du mécanisme, plutôt que de simplement acquérir un lien GSS-API sur les données existantes. De telles fonctions, si nécessaire, devraient être définies dans des sous programmes d'extension spécifiques de la mise en œuvre.

Si l'acquisition d'accréditif est consommatrice de temps pour un mécanisme, il peut choisir de retarder l'acquisition réelle jusqu'à ce que l'accréditif soit exigé (par exemple par GSS_Init_sec_context() ou GSS_Accept_sec_context()). De telles décisions de mise en œuvre spécifiques du mécanisme devraient être invisibles à l'application appelante ; donc, une invocation de GSS_Inquire_cred() suivant immédiatement l'invocation de GSS_Acquire_cred() doit retourner des données d'accréditif valides, et peut donc subir la redondance d'une acquisition d'accréditif différée.

Si GSS_C_NO_CREDENTIAL est spécifié comme input_cred_handle, un output_cred_handle non NUL doit être fourni. Pour le cas de GSS_C_NO_CREDENTIAL comme input_cred_handle, GSS_Add_cred() va créer l'accréditif référencé par

son output_cred_handle fondé sur le comportement par défaut. C'est-à-dire que l'appel aura le même effet que si l'appelant avait précédemment appelé GSS_Acquire_cred(), spécifiant le même usage et passant GSS_C_NO_NAME comme paramètre desired_name (obtenant par là un lien d'accréditif explicite correspondant au comportement par défaut), avait passé ce lien d'accréditif à GSS_Add_cred(), et avait finalement appelé GSS_Release_cred() sur le lien d'accréditif reçu de GSS_Acquire_cred().

Ce sous-programme est supposé être principalement utilisé par les accepteurs de contexte, car les mises en œuvre vont probablement fournir des façons spécifiques des mécanismes pour obtenir des accréditifs d'initiateur GSS-API de la part du processus de connexion du système. Certaines mises en œuvre peuvent donc ne pas prendre en charge l'acquisition des accréditifs GSS_C_INITIATE ou GSS_C_BOTH via GSS_Acquire_cred() pour tout nom autre que GSS_C_NO_NAME, ou un nom résultant de l'application de GSS_Inquire_context() à un contexte actif, ou un nom résultant de l'application de GSS_Inquire_cred() à un lien d'accréditif correspondant au comportement par défaut. Il est important de reconnaître que le nom explicite qui est donné en résolvant une référence par défaut peut changer au fil du temps, par exemple, par suite d'opérations de gestion d'élément d'accréditif locales en dehors de GSS-API ; une fois résolue, la valeur d'un tel nom explicite va cependant rester constante.

Un appelant peut fournir la valeur NUL (GSS_C_NO_NAME) pour le nom désiré, ce qui sera interprété comme une demande de lien d'accréditif qui va invoquer le comportement par défaut lorsqu'elle est passée à GSS_Init_sec_context(), si cred_usage est GSS_C_INITIATE ou GSS_C_BOTH, ou GSS_Accept_sec_context(), si cred_usage est GSS_C_ACCEPT ou GSS_C_BOTH.

La même entrée de nom désiré, ou de référence par défaut, devrait être utilisée sur tous les appels GSS_Acquire_cred() et GSS_Add_cred() correspondant à un accréditif particulier.

2.1.5 Invocation de GSS_Inquire_cred_by_mech

Entrées :

- o LIEN D'ACCRÉDITIF cred_handle -- si GSS_C_NO_CREDENTIAL est spécifié, les accréditifs d'initiateur par défaut sont interrogés.
- o IDENTIFIAN D'OBJET mech_type -- mécanisme spécifique pour lequel les accréditifs sont interrogés.

Sorties :

- o ENTIER major_status,
- o ENTIER minor_status,
- o NOM INTERNE cred_name, -- garanti comme étant MN ; l'appelant doit libérer avec GSS_Release_name().
- o ENTIER lifetime_rec_initiate, -- en secondes, ou valeur réservée pour INDÉFINI.
- o ENTIER lifetime_rec_accept, -- en secondes, ou valeur réservée pour INDÉFINI.
- o ENTIER cred_usage, -- 0 = INITIE-ET-ACCEPTE, 1 = INITIE-SEUL, 2 = ACCEPTE-SEUL

Codes d'état majeur en retour :

- o GSS_S_COMPLETE indique que les accréditifs référencés par l'argument d'entrée cred_handle étaient valides, que le mécanisme indiqué par l'entrée mech_type était représenté avec des éléments au sein de ces accréditifs, et que les valeurs de résultat cred_name, lifetime_rec_initiate, lifetime_rec_accept, et cred_usage représentent, respectivement, le nom de principal associé, les durées de vie restantes, et les modes d'usage convenables, des accréditifs.
- o GSS_S_NO_CRED indique qu'aucune information n'a pu être retournée sur les accréditifs référencés, soit parce que l'entrée cred_handle était invalide, soit parce que l'appelant n'a pas l'autorisation d'accéder aux accréditifs référencés.
- o GSS_S_DEFECTIVE_CREDENTIAL indique que les accréditifs référencés sont invalides.
- o GSS_S_CREDENTIALS_EXPIRED indique que les accréditifs référencés sont arrivés à expiration.
- o GSS_S_BAD_MECH indique que les accréditifs référencés ne contiennent pas d'éléments pour le mécanisme demandé.
- o GSS_S_FAILURE indique que l'opération a échoué pour des raisons non spécifiées au niveau de GSS-API.

L'invocation de GSS_Inquire_cred_by_mech() permet aux appelants dans des environnements multi mécanismes d'acquérir des données spécifiques sur les combinaisons disponibles de durée de vie, modes d'usage, et mécanismes au sein d'une structure d'accréditifs. Le résultat lifetime_rec_initiate indique la durée de vie disponible pour les besoins de l'initiation de contexte ; le résultat lifetime_rec_accept indique la durée de vie disponible pour les besoins de l'acceptation de contexte.

2.2 Invocations au niveau du contexte

Ce groupe d'appels est dédié à l'établissement et la gestion des contextes de sécurité entre les homologues. Un initiateur de contexte invoque GSS_Init_sec_context(), d'où résulte la génération d'un jeton que l'appelant passe à la cible. À la cible, ce jeton est passé à GSS_Accept_sec_context(). Selon le type de mécanisme sous-jacent et les options spécifiées, des échanges de jetons supplémentaires peuvent être effectués dans le cours de l'établissement de contexte ; de tels échanges sont traités par les retours d'état GSS_S_CONTINUE_NEEDED provenant de GSS_Init_sec_context() et GSS_Accept_sec_context().

L'une ou l'autre partie à un contexte établi peut invoquer GSS_Delete_sec_context() pour purger les informations de contexte lorsque un contexte n'est plus nécessaire. GSS_Process_context_token() est utilisé pour traiter les jetons reçus qui portent des informations de contrôle de niveau contexte. GSS_Context_time() permet à un appelant de déterminer la durée pendant laquelle un contexte établi va rester valide.

GSS_Inquire_context() retourne des informations d'état qui décrivent les caractéristiques du contexte. GSS_Wrap_size_limit() permet à un appelant de déterminer la taille d'un jeton qui sera généré par une opération GSS_Wrap(). GSS_Export_sec_context() et GSS_Import_sec_context() permettent le transfert de contextes actifs entre les processus sur un système d'extrémité.

2.2.1 Invocation de GSS_Init_sec_context

Entrées :

- LIEN D'ACCRÉDITIF claimant_cred_handle, -- NUL spécifie "utiliser la valeur par défaut".
- LIEN DE CONTEXTE input_context_handle, -- 0 (GSS_C_NO_CONTEXT) spécifie "aucun encore alloué".
- NOM INTERNE targ_name,
- IDENTIFIANT D'OBJET mech_type, -- le paramètre NUL spécifie "utiliser la valeur par défaut".
- BOOLÉEN deleg_req_flag,
- BOOLÉEN mutual_req_flag,
- BOOLÉEN replay_det_req_flag,
- BOOLÉEN sequence_req_flag,
- BOOLÉEN anon_req_flag,
- BOOLÉEN conf_req_flag,
- BOOLÉEN _req_flag,
- ENTIER, lifetime_req, -- 0 spécifie la durée de vie par défaut.
- CHAINE D'OCTETS chan_bindings,
- CHAINE D'OCTETS input_token, -- NUL ou jeton reçu de la cible.

Sorties :

- ENTIER major_status ,
- ENTIER minor_status ,
- LIEN DE CONTEXTE output_context_handle, -- une fois retourné non NUL, l'appelant doit libérer avec GSS_Delete_sec_context().
- IDENTIFIANT D'OBJET mech_type , -- le mécanisme réel est toujours indiqué, jamais NUL ; l'appelant devrait traiter comme lecture seule et ne devrait pas tenter de libérer.
- CHAINE D'OCTETS output_token , -- NUL ou jeton pour passer à la cible du contexte ; l'appelant doit libérer avec GSS_Release_buffer().
- BOOLÉEN deleg_state,
- BOOLÉEN mutual_state,
- BOOLÉEN replay_det_state,
- BOOLÉEN sequence_state,
- BOOLÉEN anon_state,
- BOOLÉEN trans_state,
- BOOLÉEN prot_ready_state, -- voir au paragraphe 1.2.7.
- BOOLÉEN conf_avail,
- BOOLÉEN integ_avail,
- ENTIER lifetime_rec -- en secondes, ou valeur réservée pour INDÉFINI.

Cet appel peut bloquer des interactions réseau en cours pou les types de mécanismes dans lesquels un serveur d'authentification ou autre entité réseau doit être consultée au nom d'un initiateur de contexte afin de générer un jeton de sortie convenable pour présentation à une cible spécifiée.

Codes d'état majeur en retour :

- GSS_S_COMPLETE indique que les informations de niveau contexte ont été initialisées avec succès, et que le jeton de résultat retourné va donner des informations suffisantes pour que la cible effectue le traitement par message sur le contexte nouvellement établi.
- GSS_S_CONTINUE_NEEDED indique que les informations de contrôle dans le jeton de résultat retourné doivent être envoyées à la cible, et qu'une réponse doit être reçue et passée comme argument de jeton de résultat à un appel de continuation à GSS_Init_sec_context(), avant que le traitement par message puisse être effectué en conjonction avec ce contexte (sauf si la valeur prot_ready_state est aussi retournée VRAI).
- GSS_S_DEFECTIVE_TOKEN indique que les vérifications de cohérence effectuées sur le jeton de résultat ont échoué, empêchant la poursuite du traitement sur la base de ce jeton.
- GSS_S_DEFECTIVE_CREDENTIAL indique que les vérifications de cohérence effectuées sur la structure d'accréditif référencée par claimant_cred_handle ont échoué, empêchant la poursuite du traitement en utilisant cette structure d'accréditif.
- GSS_S_BAD_SIG (GSS_S_BAD_MIC) indique que le jeton d'entrée reçu contient une vérification d'intégrité incorrecte, de sorte que l'établissement de contexte ne peut pas être accompli.
- GSS_S_NO_CRED indique qu'aucun contexte n'a été établi, soit parce que l'entrée cred_handle était invalide, parce que les accréditifs référencés sont valides pour l'utilisation du seul accepteur de contexte, parce que l'appelant n'a pas l'autorisation d'accéder aux accréditifs référencés, ou parce que la résolution des accréditifs par défaut a échoué.
- GSS_S_CREDENTIALS_EXPIRED indique que les accréditifs fournis par l'argument d'entrée claimant_cred_handle ne sont plus valides, de sorte que l'établissement de contexte ne peut pas être achevé.
- GSS_S_BAD_BINDINGS indique que une discordance entre le chan_binding fourni par l'appelant et ceux extraits du jeton d'entrée a été détectée, ce qui signifie un événement concernant la sécurité qui empêche l'établissement de contexte. (Ce résultat sera retourné par GSS_Init_sec_context() pour les seuls contextes où mutual_state est VRAI.)
- GSS_S_OLD_TOKEN indique que le jeton d'entrée est trop vieux pour que son intégrité soit vérifiée. C'est une erreur fatale durant l'établissement de contexte.
- GSS_S_DUPLICATE_TOKEN indique que le jeton d'entrée a une vérification d'intégrité correcte, mais est un dupliqué d'un jeton déjà traité. C'est une erreur fatale durant l'établissement de contexte.
- GSS_S_NO_CONTEXT indique qu'aucun contexte valide n'a été reconnu pour l'entrée de context_handle fournie ; cet état majeur ne sera retourné que pour les appels suivant les retours d'état GSS_S_CONTINUE_ NEEDED.
- GSS_S_BAD_NAMETYPE indique que le nom de cible fourni est d'un type non interprétable ou non accepté par le ou les mécanismes GSS-API sous-jacents applicables, de sorte que l'établissement de contexte ne peut pas être terminé.
- GSS_S_BAD_NAME indique que le nom de cible fourni est incohérent en termes d'informations de spécificateur de type incorporé en interne, de sorte que l'établissement de contexte ne peut pas être accompli.
- GSS_S_BAD_MECH indique la réception d'un jeton d'établissement de contexte ou d'une demande d'un appelant qui spécifie un mécanisme non accepté par le système local ou avec les accréditifs actifs de l'appelant.
- GSS_S_FAILURE indique que l'établissement de contexte n'a pas pu être accompli pour des raisons non spécifiées au niveau GSS-API, et qu'aucune action de récupération définie par l'interface n'est disponible.

Ce sous-programme est utilisé par un initiateur de contexte, et émet normalement un jeton de résultat convenable pour une utilisation par la cible au sein du protocole de type de mécanisme choisi. Pour le cas d'un échange à plusieurs étapes, ce jeton de résultat sera un dans une série, dont chacun est généré par un appel successif. En utilisant les informations des structures d'accréditifs référencées par claimant_cred_handle, GSS_Init_sec_context() initialise les structures de données requises pour établir un contexte de sécurité avec le targ_name cible.

Le targ_name peut être tout NOM INTERNE valide ; il n'a pas besoin d'être un MN. En plus de prendre en charge d'autres types de nom, il est recommandé (c'est une nouveauté de GSS-v2, mise à jour 1) que les mécanismes soient capables d'accepter GSS_C_NO_NAME comme type d'entrée pour targ_name. Bien que recommandé, un tel soutien n'est pas exigé, et on reconnaît que tous les mécanismes ne peuvent pas construire des jetons sans une désignation explicite de la cible du contexte, même lorsque l'authentification mutuelle de la cible n'est pas obtenue. Les appelants qui souhaitent utiliser cette facilité et sont concernés par la portabilité devraient être conscients que la prise en charge de GSS_C_NO_NAME comme type d'entrée de nom de cible ne sera vraisemblablement pas fournie au sein des définitions de mécanisme spécifiées avant GSS-v2, mise à jour 1.

Le claimant_cred_handle doit correspondre à la même structure d'accréditifs valide sur l'appel initial à GSS_Init_sec_context() et sur tout appel suivant résultant des retours d'état GSS_S_CONTINUE_NEEDED ; différentes séquences de protocole modélisées par la facilité GSS_S_CONTINUE_NEEDED vont exiger l'accès aux accréditifs à différents moments de la séquence d'établissement de contexte.

L'argument input_context_handle fourni par l'appelant doit être 0 (GSS_C_NO_CONTEXT), spécifiant "pas encore alloué", sur la première invocation de GSS_Init_sec_context() qui se rapporte à un certain contexte. Si elle réussit (c'est-à-dire, si elle est accompagnée par l'état majeur GSS_S_COMPLETE ou GSS_S_CONTINUE_NEEDED) et seulement si elle réussit, l'invocation initiale de GSS_Init_sec_context() retourne un output_context_handle différent de zéro à utiliser dans les futures références à ce contexte. Une fois qu'un output_context_handle différent de zéro a été retourné, les

appelants GSS-API devraient invoquer GSS_Delete_sec_context() pour libérer les ressources en relation avec le contexte si des erreurs surviennent dans des phases ultérieures de établissement de contexte, ou lorsque un contexte établi n'est plus nécessaire. Si GSS_Init_sec_context() est passé au lien d'un contexte qui est déjà pleinement établi, l'état GSS_S_FAILURE est retourné.

Lorsque des tentatives de continuation de GSS_Init_sec_context() sont nécessaires pour effectuer l'établissement de contexte, la valeur de lien différente de zéro précédemment retournée est entrée dans l'argument input_context_handle et il y sera fait écho dans l'argument output_context_handle retourné. Sur de telles tentatives de continuation (et seulement les tentatives de continuation) la valeur du jeton d'entrée est utilisée, pour fournir le jeton retourné de la cible du contexte.

L'argument chan_bindings est utilisé par l'appelant pour fournir des informations qui lient le contexte de sécurité aux caractéristiques qui se rapportent à la sécurité (par exemple, les adresses, les clés de chiffrement) du canal de communications sous-jacent. Voir au paragraphe 1.1.6 de ce document les détails sur l'utilisation de cet argument.

L'argument jeton_d'entrée contient un message reçu de la cible, et n'a de signification que sur une invocation de GSS_Init_sec_context() qui suit un précédent retour indiquant l'état majeur GSS_S_CONTINUE_NEEDED.

Il est de la responsabilité de l'appelant d'établir un chemin de communications vers la cible, et de transmettre tout jeton_de_retour retourné (indépendamment de la valeur d'état majeur retourné qui l'accompagne) à la cible sur ce chemin. Le jeton_de_sortie peut cependant être transmis avec le premier message d'entrée fourni par l'application pour être traité par GSS_GetMIC() ou GSS_Wrap() en conjonction avec un contexte dont l'établissement a réussi. (Note : lorsque l'indicateur GSS-v2 prot_ready_state est retourné VRAI, il est possible de transférer un message protégé avant l'achèvement de l'établissement de contexte : voir aussi le paragraphe 1.2.7).

L'initiateur peut demander diverses fonctions de niveau contexte par des fanions d'entrée : deleg_req_flag demande une délégation de droits d'accès, mutual_req_flag demande l'authentification mutuelle, replay_det_req_flag demande que le dispositif de détection de répétition soit appliqué aux messages transférés sur le contexte établi, et sequence_req_flag demande que le séquençage soit mis en application. (Voir au paragraphe 1.2.3 d'autres informations sur les dispositifs de détection de répétition et de séquençage.) anon_req_flag demande que l'identité de l'initiateur ne soit pas transférée sans que des jetons soient envoyés à l'accepteur.

Les fanions conf_req_flag et integ_req_flag fournissent des entrées d'informations pour la mise en œuvre GSS-API sur les services, respectivement, de confidentialité et d'intégrité par message, demandés sur le contexte. Ces informations sont importantes comme entrée des mécanismes de négociation. Il est important de reconnaître, cependant, que l'inclusion de ces fanions (dont la définition est une nouveauté de GSS-v2) introduit une incompatibilité arrière avec les appelants d'une mise en œuvre de GSS-v1, où ces fanions ne sont pas définis. Comme aucun appelant GSS-v1 ne va établir ces fanions, même si les services par message sont désirés, les mises en œuvre de mécanisme GSS-v2 qui activent de tels services de façon sélective sur la base des valeurs de ces fanions peuvent échouer à les fournir aux contextes établis pour les appelants GSS-v1. Il peut être donc approprié dans certaines circonstances que les mises en œuvre de tels mécanismes déduisent que ces fanions de demande de service sont établis si un appelant est connu pour être une mise en œuvre de GSS-v1.

Tous les dispositifs demandables en option ne seront pas disponibles dans tous les types de mécanisme sous-jacent. Les valeurs d'état de retour correspondantes, deleg_state, mutual_state, replay_det_state, et sequence_state indiquent, en fonction des capacités de traitement du type de mécanisme et des fanions d'entrée fournis par l'initiateur, l'ensemble des dispositifs qui seront actifs sur le contexte. La valeur trans_state retournée indique si le contexte est transférable aux autres processus par l'utilisation de GSS_Export_sec_context(). Ces valeurs d'indicateur d'état sont indéfinies sauf si l'état majeur du sous-programme indique GSS_S_COMPLETE, ou si prot_ready_state VRAI est retourné avec l'état majeur GSS_S_CONTINUE_NEEDED ; pour ce dernier cas, il est possible que des dispositifs supplémentaires, non confirmés ou indiqués avec prot_ready_state VRAI, seront confirmés et indiqués lorsque GSS_S_COMPLETE est ensuite retourné.

Les valeurs retournées anon_state et prot_ready_state sont significatives pour les deux retours d'état majeur GSS_S_COMPLETE et GSS_S_CONTINUE_NEEDED à partir de GSS_Init_sec_context(). Lorsque anon_state est retourné VRAI, cela indique que ni le jeton en cours ni ses prédécesseurs ne livrent ou n'ont livré l'identité de l'initiateur. Les appelants qui souhaitent n'effectuer l'établissement de contexte que si la prise en charge de l'anonymat est fournie devraient ne transférer un jeton retourné de GSS_Init_sec_context() à l'homologue que si il est accompagné par un indicateur anon_state VRAI. Lorsque l'état prot_ready_state est retourné VRAI en conjonction avec l'état majeur GSS_S_CONTINUE_NEEDED, cela indique que les opérations de protection par message peuvent être appliquées sur le contexte : voir au paragraphe 1.2.7 un exposé plus complet sur cette facilité.

L'échec de la fourniture de l'ensemble précis de dispositifs demandés par l'appelant ne cause pas l'échec de l'établissement de contexte ; c'est la prérogative de l'appelant de supprimer le contexte si l'ensemble de dispositifs fourni ne convient pas à l'utilisation de l'appelant.

La valeur du mech_type retournée indique le mécanisme spécifique employé sur le contexte ; il ne va jamais indiquer la valeur par "défaut". Un résultat de mech_type valide doit être retourné avec un retour d'état GSS_S_COMPLETE ; les mises en œuvre GSS-API peuvent (mais ne sont pas obligées) aussi retourner un type de mécanisme avec les appels précédents indiquant l'état GSS_S_CONTINUE_NEEDED ou (si un mécanisme peut être déterminé) en conjonction avec des cas d'erreur fatale. Dans le cas de mécanismes qui effectuent eux-mêmes la négociation, le résultat de type de mécanisme retourné peut indiquer le choix d'un mécanisme identifié par un OID différent de celui passé dans l'argument mech_type d'entrée, et la valeur retournée peut changer entre des appels successifs qui retournent GSS_S_CONTINUE_NEEDED et l'appel final qui retourne GSS_S_COMPLETE.

La valeur de retour conf_avail indique si le contexte prend en charge les services par message de confidentialité, et indique ainsi à l'appelant si une demande de chiffrement à travers l'entrée conf_req_flag de GSS_Wrap() peut ou non être honorée. De façon similaire, la valeur de retour integ_avail indique si les services d'intégrité par message sont disponibles (à travers GSS_GetMIC() ou GSS_Wrap()) sur le contexte établi. Ces valeurs d'indicateur d'état sont indéfinies sauf si l'état majeur du sous-programme indique GSS_S_COMPLETE, ou si prot_ready_state VRAI est retourné avec l'état majeur GSS_S_CONTINUE_NEEDED.

L'entrée lifetime_req spécifie une limite supérieure désirée pour la durée de vie du contexte à établir, une valeur de 0 étant utilisée pour demander la durée de vie par défaut. La valeur en retour lifetime_rec indique la durée pendant laquelle le contexte sera valide, exprimée comme un décalage par rapport au présent ; selon les capacités du mécanisme, les durées de vie des accréditifs, et la politique locale, elle peut ne pas correspondre à la valeur demandée dans lifetime_req. Si aucune contrainte n'est imposée à la durée de vie du contexte, cela peut être indiqué en retournant une valeur réservée reprenant INDÉFINI lifetime_req. La valeur de lifetime_rec est indéfinie sauf si l'état majeur du sous-programme indique GSS_S_COMPLETE.

Si mutual_state est VRAI, ce fait sera reflété au sein du jeton de résultat. Une invocation de GSS_Accept_sec_context() à la cible en conjonction avec un tel contexte va retourner un jeton, qui sera traité par un appel de continuation à GSS_Init_sec_context(), afin de réaliser l'authentification mutuelle.

2.2.2 Invocation de GSS_Accept_sec_context

Entrées :

- o LIEN D'ACCRÉDITIF acceptor_cred_handle, -- NUL spécifie "utiliser la valeur par défaut".
- o LIEN DE CONTEXTE input_context_handle, -- 0 (GSS_C_NO_CONTEXT) spécifie "non encore alloué".
- o CHAINE D'OCTETS chan_bindings,
- o CHAINE D'OCTETS input_token,

Sorties :

- o ENTIER major_status,
- o ENTIER minor_status,
- o NOM INTERNE src_name, -- garantit que c'est un MN une fois retourné, l'appelant doit libérer avec GSS_Release_name()
- o IDENTIFIANT D'OBJET mech_type, -- l'appelant devrait le traiter en lecture seule ; il n'a pas besoin de le libérer.
- o LIEN DE CONTEXTE output_context_handle, -- une fois retourné non NUL dans la séquence d'établissement de contexte, l'appelant doit libérer avec GSS_Delete_sec_context()
- o BOOLÉEN deleg_state,
- o BOOLÉEN mutual_state,
- o BOOLÉEN replay_det_state,
- o BOOLÉEN sequence_state,
- o BOOLÉEN anon_state,
- o BOOLÉEN trans_state,
- o BOOLÉEN prot_ready_state, -- voir l'exposé du paragraphe 1.2.7.
- o BOOLÉEN conf_avail,
- o BOOLÉEN integ_avail,
- o ENTIER lifetime_rec, -- en secondes, ou valeur réservée pour INDÉFINI.
- o LIEN D'ACCRÉDITIF delegated_cred_handle, -- si non NUL est retourné, l'appelant doit libérer avec GSS_Release_cred()
- o CHAINE D'OCTETS output_token, -- NULL ou jeton à passer à l'initiateur de contexte ; si le retour est non NUL, l'appelant doit libérer avec GSS_Release_buffer()

Cet appel peut bloquer les interactions réseau en instance pour les types de mécanismes dans lesquels le service d'annuaire ou autre entité réseau doit être consultée au nom d'un accepteur de contexte afin de valider un jeton d'entrée reçu.

Codes d'état majeur en retour :

- GSS_S_COMPLETE indique que les structures de données de niveau contexte ont été bien initialisées, et que le traitement par message peut maintenant être effectué en conjonction avec ce contexte.
- GSS_S_CONTINUE_NEEDED indique que les informations de contrôle dans le jeton de résultat retourné doivent être envoyées à l'initiateur, et qu'une réponse doit être reçue et passée comme l'argument du jeton d'entrée à un appel de continuation à GSS_Accept_sec_context(), avant qu'un traitement par message puisse être effectué en conjonction avec ce contexte.
- GSS_S_DEFECTIVE_TOKEN indique que les vérifications de cohérence effectuées sur le jeton d'entrée ont échoué, empêchant d'effectuer la poursuite du traitement sur la base de ce jeton.
- GSS_S_DEFECTIVE_CREDENTIAL indique que les vérifications de cohérence effectuées sur la structure d'accréditif référencée par acceptor_cred_handle ont échoué, empêchant d'effectuer la poursuite du traitement en utilisant cette structure d'accréditif.
- GSS_S_BAD_SIG (GSS_S_BAD_MIC) indique que le jeton d'entrée reçu contient une vérification d'intégrité incorrecte, de sorte que l'établissement de contexte n'a pas pu être accompli.
- GSS_S_DUPLICATE_TOKEN indique que la vérification d'intégrité sur le jeton d'entrée reçu était correcte, mais que le jeton d'entrée a été reconnu comme dupliqué d'un jeton d'entrée déjà traité. Il n'est pas établi de nouveau contexte.
- GSS_S_OLD_TOKEN indique que la vérification d'intégrité sur le jeton d'entrée reçu était correcte, mais que le jeton d'entrée est trop vieux pour qu'on fasse une vérification de duplication par rapport aux jetons d'entrée déjà traités. Il n'est pas établi de nouveau contexte.
- GSS_S_NO_CRED indique qu'il n'a pas été établi de contexte, soit parce que le cred_handle entré était invalide, parce que les accréditifs référencés ne sont valides que pour l'usage de l'initiateur de contexte, parce que l'appelant n'a pas l'autorisation d'accès aux accréditifs référencés, ou parce que la procédure de résolution d'accréditif par défaut a échoué.
- GSS_S_CREDENTIALS_EXPIRED indique que les accréditifs fournis par l'argument d'entrée acceptor_cred_handle ne sont plus valides, de sorte que l'établissement de contexte ne peut pas être achevé.
- GSS_S_BAD_BINDINGS indique qu'une discordance entre les chan_bindings fournis par l'appelant et ceux extraits du jeton d'entrée a été détectée, ce qui signifie un événement de sécurité qui empêche l'établissement de contexte.
- GSS_S_NO_CONTEXT indique qu'aucun contexte valide n'a été reconnu pour l'entrée de lien de contexte fournie ; cet état majeur ne sera retourné que pour les appels suivant le retour d'état GSS_S_CONTINUE_ NEEDED.
- GSS_S_BAD_MECH indique la réception d'un jeton d'établissement de contexte qui spécifie un mécanisme non pris en charge par le système local ou avec les accréditifs actifs de l'appelant.
- GSS_S_FAILURE indique que l'établissement de contexte n'a pas pu être accompli pour des raisons non spécifiées au niveau de GSS-API, et qu'aucune action de récupération définie par l'interface n'est disponible.

Le sous-programme GSS_Accept_sec_context() est utilisé par un contexte cible. En utilisant les informations des structures d'accréditif référencées par l'entrée acceptor_cred_handle, il vérifie le jeton d'entrée entrant et (selon la réussite de l'achèvement de la séquence d'établissement de contexte) retourne le nom de source et type de mécanisme authentifiés utilisés. Il est garanti que le nom de source retourné est un MN, traité par le mécanisme sous lequel le contexte a été établi. Le acceptor_cred_handle doit correspondre à la même structure d'accréditifs valide sur l'invocation initiale de GSS_Accept_sec_context() et sur toute invocation ultérieure résultant des retours d'état GSS_S_CONTINUE_NEEDED ; différentes séquences de protocole modélisées par le mécanisme GSS_S_CONTINUE_NEEDED vont exiger l'accès aux accréditifs à différents moments de la séquence d'établissement de contexte.

L'argument input_context_handle fourni par l'appelant doit être 0 (GSS_C_NO_CONTEXT), ce qui spécifie "non encore alloué", sur le premier appel GSS_Accept_sec_context() qui se rapporte à un certain contexte. Si il réussit (c'est-à-dire, si il est accompagné par l'état majeur GSS_S_COMPLETE ou GSS_S_CONTINUE_NEEDED), et seulement si il réussit, l'invocation initiale de GSS_Accept_sec_context() retourne un output_context_handle différent de zéro à utiliser dans les références futures à ce contexte. Une fois qu'un output_context_handle différent de zéro a été retourné, les appelants GSS-API devraient invoquer GSS_Delete_sec_context() pour libérer les ressources en rapport avec le contexte si des erreurs surviennent dans des phases ultérieures de l'établissement de contexte, ou lorsque un contexte établi n'est plus nécessaire. Si GSS_Accept_sec_context() reçoit le lien d'un contexte qui est déjà pleinement établi, l'état GSS_S_FAILURE est retourné.

L'argument chan_bindings est utilisé par l'appelant pour fournir des informations qui lient le contexte de sécurité aux caractéristiques en rapport avec la sécurité (par exemple, les adresses, les clés de chiffrement) du canal de communications sous-jacent. Voir au paragraphe 1.1.6 un exposé plus complet sur l'utilisation de cet argument.

Les résultat d'états retournés (deleg_state, mutual_state, replay_det_state, sequence_state, anon_state, trans_state, et prot_ready_state) reflètent les mêmes informations que décrit pour GSS_Init_sec_context(), et leurs valeurs ont la même signification dans les mêmes conditions de retour d'état.

Le retour de la valeur conf_avail indique si le contexte prend en charge les services de confidentialité par message, et

informe ainsi l'appelant de ce qu'une demande de chiffrement avec l'entrée de conf_req_flag dans GSS_Wrap() peut ou non être honorée. De façon similaire, le retour de la valeur integ_avail indique si les services d'intégrité par message sont disponibles (soit par GSS_GetMIC() soit par GSS_Wrap()) sur le contexte établi. Ces valeur ont la même signification que décrit sous GSS_Init_sec_context() dans les mêmes conditions d'état de retour.

Le retour de la valeur lifetime_rec n'a de signification qu'en conjonction avec l'état majeur GSS_S_COMPLETE, et indique la durée pendant laquelle le contexte sera valide, exprimée comme décalage par rapport au présent.

Le retour de la valeur mech_type indique le mécanisme spécifique employé sur le contexte ; elle ne va jamais indiquer la valeur "par défaut". Un résultat valide de mech_type doit être retourné chaque fois que l'état GSS_S_COMPLETE est indiqué ; les mises en œuvre GSS-API peuvent (mais n'y sont pas obligées) aussi retourner le mech_type avec les appels précédents qui indiquent l'état GSS_S_CONTINUE_NEEDED ou (si un mécanisme peut être déterminé) en conjonction avec des cas d'erreur fatale. Pour le cas des mécanismes qui effectuent eux-mêmes la négociation, le retour du résultat mech_type peut indiquer le choix d'un mécanisme identifié par un OID différent de celui passé dans l'argument d'entrée mech_type, et la valeur retournée peut changer entre des invocations successives qui retournent GSS_S_CONTINUE_NEEDED et l'appel final qui retourne GSS_S_COMPLETE.

Le résultat delegated_cred_handle n'a de signification que lorsque deleg_state est VRAI, et fournit à la cible le moyen de référencer les accréditifs délégués. Le jeton de sortie résultant, lorsque il n'est pas NUL, fournit un jeton de niveau contexte à retourner à l'initiateur de contexte pour continuer une séquence multi étapes d'établissement de contexte. Comme on l'a noté avec GSS_Init_sec_context(), tout jeton retourné devrait être transféré à l'homologue du contexte (dans ce cas, l'initiateur de contexte) indépendamment de la valeur de l'état majeur retourné qui l'accompagne.

Note : une cible doit être capable de distinguer un jeton d'entrée de niveau contexte qui est passé à GSS_Accept_sec_context() des éléments de données par message passés à GSS_VerifyMIC() ou GSS_Unwrap(). Ces éléments de données peuvent arriver dans un seul message d'application, et GSS_Accept_sec_context() doit être effectué avant que le traitement par message puisse être effectué avec succès.

2.2.3 Invocation de GSS_Delete_sec_context

Entrée :
- LIEN DE CONTEXTE context_handle.

Sorties :
- ENTIER major_status,
- ENTIER minor_status,
- CHAINE D'OCTETS output_context_token.

Codes d'état majeur en retour :
- GSS_S_COMPLETE indique que le contexte a été reconnu, et que les informations spécifiques de contexte pertinentes ont été purgées. Si l'appelant fournit une antémémoire non nulle pour recevoir un output_context_token, et si le mécanisme retourne un jeton non NUL dans cette antémémoire, le output_context_token retourné est prêt pour le transfert à l'homologue de contexte.
- GSS_S_NO_CONTEXT indique qu'aucun contexte valide n'a été reconnu pour l'entrée de lien de contexte fournie, de sorte qu'aucune suppression n'a été effectuée.
- GSS_S_FAILURE indique que le contexte est reconnu, mais que l'opération GSS_Delete_sec_context() n'a pas pu être effectuée pour des raisons non spécifiées au niveau GSS-API.

Cette invocation peut être faite par l'un ou l'autre homologue d'un contexte de sécurité, pour purger les informations spécifiques d'un contexte. Une fois qu'un output_context_handle différent de zéro a été retourné par les invocations d'établissement de contexte, les appelants GSS-API devraient invoquer GSS_Delete_sec_context() pour libérer les ressources qui se rapportent au contexte si des erreurs surviennent dans les phases ultérieures de l'établissement de contexte, ou lorsque un contexte établi n'est plus nécessaire. Cet appel peut bloquer des interactions réseau en instance pour les types de mécanismes dans lesquels une notification active doit être faite à un serveur central lorsque un contexte de sécurité est à supprimer.

Si un paramètre non nul output_context_token est fourni par l'appelant, un output_context_token peut être retourné à l'appelant. Si un output_context_token est fourni à l'appelant, il peut être passé à l'homologue de contexte pour informer la mise en œuvre GSS-API homologue que les informations de contexte correspondantes de l'homologue peuvent aussi être purgées. (Une fois qu'un contexte est établi, les homologues impliqués sont supposés conserver l'accréditif et les informations relatives au contexte en antémémoire jusqu'à ce que soit atteinte l'heure d'expiration des informations ou jusqu'à ce que soit faite l'invocation GSS_Delete_sec_context().)

La facilité pour l'usage du jeton de contexte de signaler la suppression de contexte est conservée pour la compatibilité avec GSS-API version 1. Pour l'usage courant, on recommande que les deux homologues d'un contexte invoquent GSS_Delete_sec_context() indépendamment, passant une antémémoire de output_context_token nulle pour indiquer qu'aucun jeton de contexte n'est nécessaire. Les mises en œuvre de GSS_Delete_sec_context() devraient supprimer les informations de contexte mémorisées en local pertinentes.

Les tentatives d'effectuer un traitement par message sur un contexte supprimé résulteront en un retour d'erreur.

2.2.4 Invocation de GSS_Process_context_token

Entrées :
- o LIEN DE CONTEXTE context_handle,
- o CHAINE D'OCTETS input_context_jeton.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que le input_context_token a été traité avec succès en conjonction avec le contexte référencé par context_handle.
- o GSS_S_DEFECTIVE_TOKEN indique que les vérifications de cohérence effectuées sur le jeton de contexte reçu ont échoué, empêchant d'effectuer la poursuite du traitement avec ce jeton.
- o GSS_S_NO_CONTEXT indique qu'aucun contexte valide n'a été reconnu pour l'entrée de context_handle fournie.
- o GSS_S_FAILURE indique que le contexte est reconnu, mais que l'opération GSS_Process_context_token() n'a pas pu être effectuée pour des raisons non spécifiées au niveau GSS-API.

Cette invocation est utilisée pour traiter les jetons de contexte reçus d'un homologue une fois qu'un contexte a été établi, avec l'impact correspondant sur les informations d'état de niveau contexte. Une utilisation de cette facilité est le traitement des jetons de contexte générés par GSS_Delete_sec_context() ; à cette fin, GSS_Process_context_token() ne va pas bloquer les interactions réseau en instance. Une autre utilisation est de traiter les jetons qui indiquent les défaillances d'établissement de contexte chez l'homologue distant après le moment où la mise en œuvre GSS-API locale a déjà indiqué l'état GSS_S_COMPLETE.

2.2.5 Invocation de GSS_Context_time

Entrée :
- o LIEN DE CONTEXTE context_handle,

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o ENTIER lifetime_rec -- en secondes, ou valeur réservée pour INDÉFINI.

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que le contexte référencé est valide, et va rester valide pendant la durée indiquée dans lifetime_rec.
- o GSS_S_CONTEXT_EXPIRED indique que les éléments de données qui se rapportent au contexte référencé sont périmés.
- o GSS_S_NO_CONTEXT indique qu'aucun contexte valide n'a été reconnu pour l'entrée de lien de contexte fournie.
- o GSS_S_FAILURE indique que l'opération demandée a échoué pour des raisons non spécifiées au niveau GSS-API.

Cet appel est utilisé pour déterminer la durée pendant laquelle un contexte actuellement établi va rester valide.

2.2.6 Invocation de GSS_Inquire_context

Entrée :
- o LIEN DE CONTEXTE context_handle,

Sorties :
- o ENTIER major_status,

- o ENTIER minor_status,
- o NOM INTERNE src_name, -- nom de l'initiateur de contexte, garanti d'être MN ; l'appelant doit libérer avec GSS_Release_name() si il est retourné.
- o NOM INTERNE targ_name, -- nom de la cible du contexte, garanti d'être MN; l'appelant doit libérer avec GSS_Release_name() si il est retourné.
- o ENTIER lifetime_rec -- en secondes, ou valeur réservée pour INDÉFINI ou EXPIRÉ.
- o IDENTIFIANT D'OBJET mech_type, -- mécanisme qui prend en charge ce contexte de sécurité ; l'appelant devrait le traiter comme en lecture seule et ne pas tenter de libérer.
- o BOOLÉEN deleg_state,
- o BOOLÉEN mutual_state,
- o BOOLÉEN replay_det_state,
- o BOOLÉEN sequence_state,
- o BOOLÉEN anon_state,
- o BOOLÉEN trans_state,
- o BOOLÉEN prot_ready_state,
- o BOOLÉEN conf_avail,
- o BOOLÉEN integ_avail,
- o BOOLÉEN locally_initiated, -- VRAI si c'est l'initiateur, FAUX si c'est l'accepteur.
- o BOOLÉEN open, -- VRAI si le contexte est pleinement établi, FAUX si il est partiellement établi (dans l'état CONTINUE_NEEDED)

Codes d'état majeur en retour :

- o GSS_S_COMPLETE indique que le contexte référencé est valide et que les valeurs retournées de deleg_state, mutual_state, replay_det_state, sequence_state, anon_state, trans_state, prot_ready_state, conf_avail, integ_avail, locally_initiated, et open décrivent les caractéristiques correspondantes du contexte. Si open est VRAI, lifetime_rec est aussi retourné : si open est VRAI et si le nom de l'homologue du contexte est connu, src_name et targ_name sont valides en plus des valeurs énumérées ci-dessus. La valeur de mech_type doit être retournée pour les contextes où open est VRAI et peut être retournée pour les contextes où open est FAUX.
- o GSS_S_NO_CONTEXT indique qu'aucun contexte valide n'a été reconnu pour l'entrée context_handle fournie. Les valeurs retournées autres que major_status et minor_status sont indéfinies.
- o GSS_S_FAILURE indique que l'opération demandée a échoué pour des raisons non spécifiées au niveau GSS-API. Les valeurs retournées autres que major_status et minor_status sont indéfinies.

Cet appel est utilisé pour extraire les informations qui décrivent les caractéristiques d'un contexte de sécurité. Noter que les mises en œuvre GSS-API sont supposées conserver les données de contexte interrogeables jusqu'à ce que le contexte soit libéré par un appelant, même après l'expiration du contexte, bien que les éléments de données cryptographiques sous-jacents puissent être supprimés après l'expiration afin de limiter leur exposition.

2.2.7 Invocation de GSS_Wrap_size_limit

Entrées :

- o LIEN DE CONTEXTE context_handle,
- o BOOLÉEN conf_req_flag,
- o ENTIER qop,
- o ENTIER output_size.

Sorties :

- o ENTIER major_status,
- o ENTIER minor_status,
- o ENTIER max_input_size.

Codes d'état majeur en retour :

- o GSS_S_COMPLETE indique la réussite de la détermination de la taille d'un jeton : un message d'entrée avec une longueur en octets égale à la valeur retournée de max_input_size va, lorsque elle est passée à GSS_Wrap() pour traitement sur le contexte identifié par le paramètre context_handle avec l'état de demande de confidentialité comme donnée dans le conf_req_flag et avec le spécificateur de qualité de protection fourni dans le paramètre qop, donne un jeton de sortie qui n'est pas plus grand que la valeur du paramètre output_size fourni.
- o GSS_S_CONTEXT_EXPIRED indique que l'entrée de input context_handle fournie est reconnue, mais que le contexte référencé est périmé. Les valeurs retournées autres que major_status et minor_status sont indéfinies.
- o GSS_S_NO_CONTEXT indique qu'aucun contexte valide n'a été reconnu pour l'entrée de context_handle fournie. Les valeurs retournées autres que major_status et minor_status sont indéfinies.
- o GSS_S_BAD_QOP indique que la valeur de QOP fournie n'est pas reconnue ou prise en charge par le contexte.

- GSS_S_FAILURE indique que l'opération demandée a échoué pour des raisons non spécifiées au niveau GSS-API. Les valeurs retournées autres que major_status et minor_status sont indéfinies.

Cet appel est utilisé pour déterminer la plus grande quantité de données d'entrée qui peut être passée à GSS_Wrap() sans donner un jeton de sortie plus grand qu'une valeur spécifiée par l'appelant.

2.2.8 Invocation de GSS_Export_sec_context

Entrées :
- LIEN DE CONTEXTE context_handle.

Sorties :
- ENTIER major_status,
- ENTIER minor_status,
- CHAINE D'OCTETS interprocess_token -- l'appelant doit libérer avec GSS_Release_buffer().

Codes d'état majeur en retour :
- GSS_S_COMPLETE indique que le contexte référencé a été bien exporté à une représentation dans le interprocess_token, et n'est plus disponible pour être utilisé par l'appelant.
- GSS_S_UNAVAILABLE indique que la facilité d'export du contexte n'est pas disponible pour utilisation sur le contexte référencé. (Cet état ne devrait survenir que pour les contextes pour lesquels la valeur trans_state est FAUX.) Les valeurs de retour autres que major_status et minor_status sont indéfinies.
- GSS_S_CONTEXT_EXPIRED indique que le lien de contexte fourni en entrée est reconnu, mais que le contexte référencé est périmé. Les valeurs de retour autres que major_status et minor_status sont indéfinies.
- GSS_S_NO_CONTEXT indique qu'aucun contexte valide n'a été reconnu pour le lien de contexte fourni en entrée. Les valeurs de retour autres que major_status et minor_status sont indéfinies.
- GSS_S_FAILURE indique que l'opération demandée a échoué pour des raisons non spécifiées au niveau de GSS-API. Les valeurs de retour autres que major_status et minor_status sont indéfinies.

Cet appel génère un jeton interprocessus à transférer à un autre processus au sein d'un système d'extrémité, afin de transférer le contrôle d'un contexte de sécurité à ce procès. Le receveur du jeton interprocès va invoquer GSS_Import_sec_context() pour accepter le transfert. L'opération GSS_Export_sec_context() est définie pour le seul usage avec les contextes de sécurité qui sont pleinement établis avec succès (c'est-à-dire, ceux pour lesquels GSS_Init_sec_context() et GSS_Accept_sec_context() ont retourné l'état majeur GSS_S_COMPLETE).

Une opération GSS_Export_sec_context() réussie désactive le contexte de sécurité pour le processus appelant ; pour ce cas, la mise en œuvre GSS-API devra désallouer toutes les ressources associées au niveau du processus au contexte de sécurité et devra régler le context_handle à GSS_C_NO_CONTEXT. Dans le cas d'une erreur qui rend impossible d'achever l'exportation du contexte de sécurité, la mise en œuvre GSS-API ne doit pas retourner un jeton interprocès et devrait amener la mise en œuvre à laisser le contexte de sécurité référencé par le lien de contexte inchangé. Si c'est impossible, il est permis à la mise en œuvre de supprimer le contexte de sécurité, tout en réglant aussi le paramètre context_handle à GSS_C_NO_CONTEXT.

Les appelants portables ne doivent pas supposer qu'un certain jeton interprocès peut être importé plus d'une fois par GSS_Import_sec_context(), créant par là plusieurs instanciations d'un seul contexte. Les mises en œuvre GSS-API peuvent détecter et rejeter les tentatives d'importation multiples, mais ne sont pas obligées de le faire.

La représentation interne contenue dans le jeton interprocès est une affaire locale définie par la mise en œuvre. Les jetons interprocès ne peuvent pas être supposés transférables à travers différentes mises en œuvre GSS-API.

Il est recommandé que les mises en œuvre GSS-API adoptent des politiques adaptées à leur environnements de fonctionnement afin de définir l'ensemble des processus éligibles pour importer un contexte, mais les contraintes spécifiques de ce domaine sont des affaires locales. Des exemples possibles incluent les transferts entre processus fonctionnant au nom de la même identité d'usager, ou des processus qui comportent des tâches communes. Cependant, il peut être impossible de mettre en application de telles politiques dans certaines mises en œuvre.

À l'appui des buts ci-dessus, les mises en œuvre peuvent protéger les données du contexte transféré en utilisant le chiffrement pour protéger les données au sein du jeton interprocès, ou en utilisant les jetons interprocès comme moyen de faire référence à des facilités locales de communication interprocès (protégées par d'autres moyens) plutôt que de mémoriser directement les données du contexte dans le jeton.

Le transfert d'un contexte ouvert peut, pour certains mécanismes et mises en œuvre, révéler des données sur l'accréditif qui

a été utilisé pour établir le contexte. Les appelants devraient donc faire attention à la confiance qu'ils peuvent accorder aux processus auxquels ils transfèrent les contextes. Bien que la mise en œuvre GSS-API puisse fournir son propre ensemble de protections sur le contexte exporté, l'appelant est responsable de la protection du jeton interprocès contre la divulgation, et de prendre soin que le contexte soit transféré à un processus de destination approprié.

2.2.9 Invocation de GSS_Import_sec_context

Entrées :
- o CHAINE D'OCTETS interprocess_token.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o LIEN DE CONTEXTE context_handle. -- si il est bien retourné, l'appelant doit libérer avec GSS_Delete_sec_context().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que le contexte représenté par l'entrée interprocess_token a été bien transféré à l'appelant, et est disponible pour utilisation future via le context_handle résultant.
- o GSS_S_NO_CONTEXT indique que le contexte représenté par l'entrée interprocess_token était invalide. Les valeurs de retour autres que major_status et minor_status sont indéfinies.
- o GSS_S_DEFECTIVE_TOKEN indique que l'entrée de interprocess_token était défectueuse. Les valeurs de retour autres que major_status et minor_status sont indéfinies.
- o GSS_S_UNAVAILABLE indique que la facilité d'importation du contexte n'est pas disponible pou utilisation sur le contexte référencé. Les valeurs de retour autres que major_status et minor_status sont indéfinies.
- o GSS_S_UNAUTHORIZED indique que le contexte représenté par l'entrée de interprocess_token n'est pas autorisée pour le transfert à l'appelant. Les valeurs de retour autres que major_status et minor_status sont indéfinies.
- o GSS_S_FAILURE indique que l'opération a échoué pour des raisons non spécifiées au niveau GSS-API. Les valeurs de retour autres que major_status et minor_status sont indéfinies.

Ce appel traite un jeton interprocessus généré par GSS_Export_sec_context(), rendant le contexte transféré disponible à l'utilisation par l'appelant. Après le succès d'une opération GSS_Import_sec_context(), le contexte importé est disponible pour être utilisé par le processus importateur. En particulier, le contexte importé est utilisable pour toutes les opérations par message et peut être supprimé ou exporté par son importateur. L'incapacité à recevoir des accréditifs délégués par gss_import_sec_context() empêche l'établissement de nouveaux contextes sur la base des informations déléguées au système d'extrémité de l'importateur au sein du contexte qui est importé, sauf si ces accréditifs délégués sont obtenus par des sous-programmes séparés (par exemple, des invocations de XGSS-API) en dehors de la définition de GSS-v2.

Pour un exposé complémentaires sur les questions de sécurité et d'autorisation concernant cet appel, voir au paragraphe 2.2.8.

2.3 Invocations par message

Ce groupe d'appels est utilisé pour effectuer le traitement de protection par message sur un contexte de sécurité établi. Aucun de ces appels ne bloque les interactions réseau en instance. Ces appels peuvent être invoqués par l'initiateur d'un contexte ou par la cible d'un contexte. Les quatre membres de ce groupe devraient être considérés comme deux paires ; le résultat de GSS_GetMIC() est entré dans GSS_VerifyMIC(), et le résultat de GSS_Wrap() est entré dans GSS_Unwrap().

GSS_GetMIC() et GSS_VerifyMIC() prennent en charge les services d'authentification d'origine des données et de protection de l'intégrité. Lorsque GSS_GetMIC() est invoqué sur un message d'entrée, il donne un jeton par message qui contient des éléments de données qui permettent aux mécanismes sous-jacents de fournir les services de sécurité spécifiés. Le message d'origine, avec le jeton par message généré, est passé à l'homologue distant ; ces deux éléments de données sont traités par GSS_VerifyMIC(), qui valide le message en conjonction avec le jeton séparé.

GSS_Wrap() et GSS_Unwrap() prennent en charge la confidentialité demandée par l'appelant en plus des services d'authentification d'origine des données et de protection de l'intégrité offerts par GSS_GetMIC() et GSS_VerifyMIC(). GSS_Wrap() sort un seul élément de données, encapsulant facultativement les données chiffrées d'usager ainsi que les éléments de données du jeton associé. L'élément de données résultant de GSS_Wrap() est passé à l'homologue distant et traité par GSS_Unwrap() chez ce système. GSS_Unwrap() combine le déchiffrement (en tant que de besoin) avec la validation des éléments de données qui se rapportent à l'authentification et l'intégrité.

Bien que les jetons de longueur zéro ne soient jamais retournés par les appels GSS pour être transférés à l'homologue d'un contexte, un objet de longueur zéro peut être passé par un appelant dans GSS_Wrap(), auquel cas, l'homologue correspondant invoquant GSS_Unwrap() sur le jeton transféré va recevoir un objet de longueur zéro comme résultat de GSS_Unwrap(). De même, GSS_GetMIC() peut être invoqué sur un objet vide, donnant un MIC que GSS_VerifyMIC() va réussir à vérifier par rapport au contexte de sécurité actif en conjonction avec un objet de longueur zéro.

2.3.1 Invocation de GSS_GetMIC

Note : Cet appel est fonctionnellement équivalent à l'invocation de GSS_Sign comme défini dans les précédentes versions de cette spécification. Dans l'intérêt de la rétro compatibilité, il est recommandé que les mises en œuvre prennent en charge pour le présent cette fonction sous les deux noms ; les références futures à cette fonction sous le nom de GSS_Sign sont déconseillées.

Entrées :
- o LIEN DE CONTEXTE context_handle,
- o ENTIER op_req, -- 0 spécifie la QOP par défaut.
- o CHAINE D'OCTETS message.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o CHAINE D'OCTETS per_msg_token. -- l'appelant doit libérer avec GSS_Release_buffer().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique qu'une vérification d'intégrité, convenable pour un contexte de sécurité établi, a été appliquée avec succès et que le message et le per_msg_token correspondant sont prêts pour la transmission.
- o GSS_S_CONTEXT_EXPIRED indique que les éléments de données en rapport avec le contexte sont périmés, de sorte que l'opération demandée ne peut pas être effectuée.
- o GSS_S_NO_CONTEXT indique qu'aucun contexte n'a été reconnu pour l'entrée de context_handle fournie.
- o GSS_S_BAD_QOP indique que la valeur de QOP fournie n'est pas reconnue ou prise en charge pour le contexte.
- o GSS_S_FAILURE indique que le contexte est reconnu, mais que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

L'utilisation du contexte de sécurité référencé par context_handle, applique une vérification d'intégrité au message d'entrée (avec les horodatages et/ou autres données incluses en soutien des mécanismes spécifiques du mech_type) et (si l'état GSS_S_COMPLETE est indiqué) retourne le résultat dans per_msg_token. Le paramètre qop_req, dont l'interprétation est exposée au paragraphe 1.2.4, permet le contrôle de la qualité de protection. L'appelant passe le message et le per_msg_token à la cible.

La fonction GSS_GetMIC() s'achève avant que le message et le per_msg_token soient envoyés à l'homologue ; l'application réussie de GSS_GetMIC() ne garantit pas qu'un GSS_VerifyMIC() correspondant a été (ou peut nécessairement être) effectué avec succès lorsque le message arrive à destination.

Les mécanismes qui ne prennent pas en charge les services de protection par message devraient retourner GSS_S_FAILURE si ce sous-programme est invoqué.

2.3.2 Invocation de GSS_VerifyMIC

Note : Cet appel est fonctionnellement équivalent à l'invocation de GSS_Verify comme défini dans les précédentes versions de cette spécification. Dans l'intérêt de la rétro compatibilité, il est recommandé que les mises en œuvre prennent pour l'heure en charge cette fonction sous les deux noms ; à l'avenir, les références à cette fonction sous le nom de GSS_Verify sont déconseillées.

Entrées :
- o LIEN DE CONTEXTE context_handle,
- o CHAINE D'OCTETS message,
- o CHAINE D'OCTETS per_msg_token.

Sorties :
- o ENTIER qop_state,
- o ENTIER major_status,
- o ENTIER minor_status,

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que le message a été bien vérifié.
- o GSS_S_DEFECTIVE_TOKEN indique que les vérifications de cohérence effectuées sur le jeton par message reçu ont échoué, empêchant d'effectuer la poursuite du traitement avec ce jeton.
- o GSS_S_BAD_SIG (GSS_S_BAD_MIC) indique que le jeton par message reçu contient une vérification d'intégrité incorrecte pour le message.
- o Les valeurs de GSS_S_DUPLICATE_TOKEN, GSS_S_OLD_TOKEN, GSS_S_UNSEQ_TOKEN, et GSS_S_GAP_TOKEN apparaissent en conjonction avec les dispositifs facultatifs de détection de répétition par message décrits au paragraphe 1.2.3 ; leur sémantique est décrite dans cette section.
- o GSS_S_CONTEXT_EXPIRED indique que les éléments de données en rapport avec le contexte sont périmés, de sorte que l'opération demandée ne peut pas être effectuée.
- o GSS_S_NO_CONTEXT indique qu'aucun contexte n'a été reconnu pour l'entrée de lien de contexte fournie.
- o GSS_S_FAILURE indique que le contexte est reconnu, mais que l'opération GSS_VerifyMIC() n'a pas pu être effectuée pour des raisons non spécifiées au niveau GSS-API.

En utilisant le contexte de sécurité référencé par context_handle, on vérifie que l'entrée de jeton par message contient une vérification d'intégrité appropriée pour le message d'entrée, et on applique tout dispositif de détection active de répétition ou de séquençage. Il retourne une indication de la qualité de protection appliquée au message traité dans le qop_state résultant.

Les mécanismes qui ne prennent pas en charge les services de protection par message devraient retourner GSS_S_FAILURE si ce sous-programme est invoqué.

2.3.3 Invocation de GSS_Wrap

Note : Cet appel est fonctionnellement équivalent à l'invocation de GSS_Seal comme défini dans les précédentes versions de cette spécification. Dans l'intérêt de la rétro compatibilité, il est recommandé que les mises en œuvre prennent pour l'heure en charge cette fonction sous les deux noms ; à l'avenir, les références à cette fonction sous le nom de GSS_Seal sont déconseillées.

Entrées :
- o LIEN DE CONTEXTE context_handle,
- o BOOLÉEN conf_req_flag,
- o ENTIER qop_req, -- 0 spécifie la QOP par défaut.
- o CHAINE D'OCTETS input_message.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o BOOLÉEN conf_state,
- o CHAINE D'OCTETS output_message -- l'appelant doit libérer avec GSS_Release_buffer().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que le traitement du message d'entrée a réussi et que le message de sortie est prêt à la transmission.
- o GSS_S_CONTEXT_EXPIRED indique que les éléments de données en rapport avec le contexte sont périmés, de sorte que l'opération demandée ne peut pas être effectuée.
- o GSS_S_NO_CONTEXT indique qu'aucun contexte n'a été reconnu pour l'entrée de lien de contexte fournie.
- o GSS_S_BAD_QOP indique que la valeur de QOP fournie n'est pas reconnue ou prise en charge pour le contexte.
- o GSS_S_FAILURE indique que le contexte est reconnu, mais que l'opération GSS_Wrap() n'a pas pu être effectuée pour des raisons non spécifiées au niveau GSS-API.

Effectue les fonctions d'authentification d'origine des données et d'intégrité des données de GSS_GetMIC(). Si l'entrée de conf_req_flag est VRAI, il demande que la confidentialité soit appliquée au message d'entrée. La confidentialité peut n'être pas prise en charge dans tous les types de mécanismes ou par toutes les mises en œuvre ; le retour du fanion conf_state indique si la confidentialité était fournie pour le message d'entrée. Le paramètre qop_req, dont l'interprétation est exposée au paragraphe 1.2.4, permet le contrôle de la qualité de protection.

Lorsque l'état GSS_S_COMPLETE est retourné, l'invocation de GSS_Wrap() donne un seul éléments de données de message de sortie (facultativement chiffré) contenant les données d'utilisateur ainsi que les informations de contrôle.

Les mécanismes qui ne prennent pas en charge les services de protection par message devraient retourner GSS_S_FAILURE si ce sous-programme est invoqué.

2.3.4 Invocation de GSS_Unwrap

Note : Cet appel est fonctionnellement équivalent à l'invocation de GSS_Unseal comme défini dans les précédentes versions de cette spécification. Dans l'intérêt de la rétro compatibilité, il est recommandé que les mises en œuvre prennent pour l'heure en charge cette fonction sous les deux noms ; à l'avenir, les références à cette fonction sous le nom de GSS_Unseal sont déconseillées.

Entrées :
- o LIEN DE CONTEXTE context_handle,
- o CHAINE D'OCTETS input_message.

Sorties :
- o BOOLÉEN conf_state,
- o ENTIER qop_state,
- o ENTIER major_status,
- o ENTIER minor_status,
- o CHAINE D'OCTETS output_message -- l'appelant doit libérer avec GSS_Release_buffer().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que le traitement du message d'entrée a réussi et que le message de sortie résultant est disponible.
- o GSS_S_DEFECTIVE_TOKEN indique que les vérifications de cohérence effectuées sur le jeton par message extrait du message d'entrée ont échoué, empêchant d'effectuer la poursuite du traitement.
- o GSS_S_BAD_SIG (GSS_S_BAD_MIC) indique qu'une vérification d'intégrité incorrecte a été détectée pour le message.
- o Les valeurs GSS_S_DUPLICATE_TOKEN, GSS_S_OLD_TOKEN, GSS_S_UNSEQ_TOKEN, et GSS_S_GAP_TOKEN apparaissent en conjonction avec le dispositif facultatif de détection de répétition par message décrit au paragraphe 1.2.3 ; leur sémantique est décrite dans cette section.
- o GSS_S_CONTEXT_EXPIRED indique que les éléments de données en rapport avec le contexte sont périmés, de sorte que l'opération demandée ne peut pas être effectuée.
- o GSS_S_NO_CONTEXT indique qu'aucun contexte n'a été reconnu pour l'entrée de lien de contexte fournie.
- o GSS_S_FAILURE indique que le contexte est reconnu mais que l'opération GSS_Unwrap() n'a pas pu être effectuée pour des raisons non spécifiées au niveau GSS-API.

Traite un élément de données généré (et facultativement chiffré) par GSS_Wrap(), fourni comme message d'entrée. La valeur de conf_state retournée indique si la confidentialité a été appliquée au message d'entrée.
Si conf_state est VRAI, GSS_Unwrap() a déchiffré le message d'entrée. Il retourne une indication de la qualité de protection appliquée au message traité dans le qop_state résultant. GSS_Unwrap() effectue les fonctions de vérification d'intégrité des données et d'authentification d'origine des données de GSS_VerifyMIC() sur les données du texte en clair. Les données du texte en clair sont retournées dans output_message.

Les mécanismes qui ne prennent pas en charge les services de protection par message devraient retourner GSS_S_FAILURE si ce sous-programme est invoqué.

2.4 Invocations de soutien

Ce groupe d'appels fournit des fonctions de soutien utiles pour les appelants GSS-API, indépendantes de l'état des contextes établis. Leur caractérisation par rapport à l'état de blocage ou non blocage en termes d'interactions du réseau n'est pas spécifiée.

2.4.1 Invocation de GSS_Display_status

Entrées :
- o ENTIER status_value, -- valeur en retour d'état majeur ou d'état mineur GSS-API.
- o ENTIER status_type, -- 1 si état majeur, 2 si état mineur.
- o IDENTIFIANT D'OBJET mech_type -- type de mécanisme à utiliser pour la traduction d'état mineur.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o ENSEMBLE DE CHAINE D'OCTETS status_string_set -- les appels requis pour la libération par l'appelant sont spécifiques des liens de langage.

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique qu'une représentation valide d'état imprimable (représentant éventuellement plus d'un événement d'état codé au sein de status_value) est disponible dans le status_string_set retourné.
- o GSS_S_BAD_MECH indique que la traduction selon un mech_type non pris en charge a été demandée, de sorte que la traduction n'a pas pu être effectuée.
- o GSS_S_BAD_STATUS indique que l'entrée de status_value était invalide, ou que l'entrée status_type portait une valeur autre que 1 ou 2, de sorte que la traduction n'a pas pu être effectuée.
- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Donne aux appelants un moyen de traduire les codes d'état majeur et mineur GSS-API retournés en représentations de chaîne imprimable.

Note : certains liens de langage peuvent employer une approche itérative afin d'émettre des composant d'état successifs ; cette approche est acceptable mais pas exigée pour la conformité à la spécification actuelle.

Bien que ce ne soit pas dans la [RFC2078], il a été observé que certaines mises en œuvre GSS-API existantes retournent l'état GSS_S_CONTINUE_NEEDED lors d'une itération à travers des messages successifs retournés de GSS_Display_status(). Ce comportement est déconseillé ; GSS_S_CONTINUE_NEEDED devrait n'être retourné que par GSS_Init_sec_context() et GSS_Accept_sec_context(). Pour une portabilité maximale, il est cependant recommandé que l'appelant prudent soit capable d'accepter et d'ignorer l'état GSS_S_CONTINUE_NEEDED si il est indiqué par GSS_Display_status() ou tout appel autre que GSS_Init_sec_context() ou GSS_Accept_sec_context().

2.4.2 Invocation de GSS_Indicate_mechs

Entrée :
- o (aucune)

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o ENSEMBLE D'IDENTIFIANTS D'OBJET mech_set -- l'appelant doit libérer avec GSS_Release_oid_set().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique qu'un ensemble de mécanismes disponibles a été retourné dans mech_set.
- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Permet à l'appelant de déterminer l'ensemble de types de mécanisme disponible sur le système local. Cet appel est destiné à la prise en charge d'appelants spécialisés qui ont besoin de demander des ensembles de types de mécanismes qui ne sont pas par défaut à des invocations de GSS-API qui acceptent des entrées de spécificateur de type de mécanisme.

2.4.3 Invocation de GSS_Compare_name

Entrées :
- o NOM INTERNE name1,
- o NOM INTERNE name2.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o BOOLÉEN name_equal.

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que name1 et name2 étaient comparables, et le résultat name_equal indique si name1 et name2 représentent la même entité.
- o GSS_S_BAD_NAMETYPE indique que les deux entrées de types de nom sont différentes et incomparables, de sorte

que l'opération de comparaison n'a pas pu être achevée.
- o GSS_S_BAD_NAME indique que un des noms d'entrée ou les deux étaient mal formés en termes de spécificateur de type interne, de sorte que l'opération de comparaison n'a pas pu être achevée.
- o GSS_S_FAILURE indique que l'opération d'appel n'a pas pu être effectuée pour des raisons non spécifiées au niveau GSS-API.

Permet aux appelants de comparer deux représentations de nom interne pour déterminer si elles se réfèrent à la même entité. Si un des noms présentés à GSS_Compare_name() note un principal anonyme, GSS_Compare_name() devra indiquer FAUX. Il n'est pas exigé que l'un ou l'autre ou les deux name1 et name2 entrés soient des MN ; pour certaines mises en œuvre et dans certains cas, GSS_S_BAD_NAMETYPE peut être retourné, ce qui indique que les noms ne sont pas comparables, pour le cas où aucun des noms entrés n'est un MN.

2.4.4 Invocation de GSS_Display_name

Entrées :
- o NOM INTERNE name.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o CHAINE D'OCTETS name_string, -- l'appelant doit libérer avec GSS_Release_buffer().
- o IDENTIFIANT D'OBJET name_type -- l'appelant devrait le traiter en lecture seule ; n'a pas besoin d'être libéré.

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique qu'une représentation valide de nom imprimable est disponible dans la name_string retournée.
- o GSS_S_BAD_NAME indique que le contenu du nom fourni n'était pas cohérent avec le type de nom indiqué en interne, de sorte qu'aucune représentation imprimable n'a pu être générée.
- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Permet aux appelants de traduire une représentation de nom interne en forme imprimable avec le descripteur de type d'espace de nom associé. La syntaxe de la forme imprimable est une affaire locale.

Si l'entrée de nom représente une identité anonyme, une valeur réservée (GSS_C_NT_ANONYMOUS) devra être retournée pour name_type.

Le type de nom GSS_C_NO_OID n'est à retourner que lorsque le nom interne correspondant a été créé par une importation avec GSS_C_NO_OID. Il est acceptable pour les mécanismes de normaliser les noms importés avec GSS_C_NO_OID en d'autres types pris en charge et, donc, de les afficher avec des types autres que GSS_C_NO_OID.

2.4.5 Invocation de GSS_Import_name

Entrées :
- o CHAINE D'OCTETS input_name_string,
- o IDENTIFIANT D'OBJET input_name_type.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o NOM INTERNE output_name -- l'appelant doit libérer avec GSS_Release_name().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique qu'une représentation valide de nom est sortie dans output_name et est décrite par la valeur de type dans output_name_type.
- o GSS_S_BAD_NAMETYPE indique que l'entrée input_name_type n'est pas acceptée par le ou les mécanismes GSS-API sous-jacents applicables, de sorte que l'opération d'importation n'a pas pu être achevée.
- o GSS_S_BAD_NAME indique que la input_name_string fournie est mal formée en termes de input_name_type, de sorte que l'opération d'importation n'a pas pu être achevée.
- o GSS_S_BAD_MECH indique que l'entrée présentée pour être importée était un objet Nom exporté et que son type de mécanisme inclus n'a pas été reconnu ou n'était pas accepté par la mise en œuvre GSS-API.

- GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Permet aux appelants de fournir une représentation de nom comme une chaîne d'octets contigus, de désigner le type d'espace de noms en conjonction avec lequel il devrait être analysé, et de convertir cette représentation en une forme interne convenable pour l'entrée dans les autres sous-programmes GSS-API. La syntaxe de input_name_string est définie en conjonction avec son type de nom associé ; selon le input_name_type, la chaîne associée input_name_string peut être ou non une chaîne imprimable. Si la valeur du input_name_type est GSS_C_NO_OID, une syntaxe imprimable par défaut spécifique du mécanisme (qui devra être spécifiée dans la spécification correspondante de mécanisme GSS-v2) est supposée pour la input_name_string ; d'autres valeurs de input_name_type enregistrées par les mises en œuvre GSS-API peuvent être utilisées pour indiquer des syntaxes de nom spécifiques autres que par défaut.

Note : l'argument input_name_type sert à décrire et qualifier l'interprétation de la input_name_string associée ; il ne spécifie pas le type de données du output_name retourné.

Si un mécanisme revendique la prise en charge d'un type de nom particulier, son opération GSS_Import_name() devra être capable d'accepter toutes les valeurs possibles conformes à la syntaxe de nom externe comme définie pour ce type de nom. Ces valeurs importées peuvent correspondre à :
(1) des entités enregistrées en local (pour lesquelles des accréditifs peuvent être acquis)
(2) des entités non locales (pour lesquelles des accréditifs locaux ne peuvent pas être acquis, mais qui peuvent être référencés comme cibles des contextes de sécurité initiés ou comme initiateurs de contextes de sécurité acceptés)
(3) aucun des deux.

Déterminer si un nom particulier appartient à la classe (1), (2), ou (3) décrites ci-dessus ne garantit qu'il puisse être traité par la fonction GSS_Import_name().

Le nom interne généré par une opération GSS_Import_name() peut être un MN d'un seul mécanisme, et sera probablement un MN au sein d'une mise en œuvre d'un seul mécanisme, mais les appelants portables ne doivent pas s'appuyer sur cette propriété (et ne doivent donc pas supposer que le résultat de GSS_Import_name() peut être passé directement à GSS_Export_name() sans être d'abord traité par GSS_Canonicalize_name()).

2.4.6 Invocation de GSS_Release_name

Entrées :
- NOM INTERNE name.

Sorties :
- ENTIER major_status,
- ENTIER minor_status.

Codes d'état majeur en retour :
- GSS_S_COMPLETE indique que la mémorisation associée à l'entrée de nom a bien été libérée.
- GSS_S_BAD_NAME indique que l'argument de nom entré ne contenait pas un nom valide.
- GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Permet à l'appelant de libérer la mémorisation associée à une représentation de nom interne. Ce comportement spécifique de l'appelant dépend de l'environnement de langage et de programmation au sein duquel fonctionne une mise en œuvre GSS-API, et est donc détaillé au sein des spécifications de liens applicables ; en particulier, la mise en œuvre et l'invocation de cet appel peut être superflue (et peut être omise) au sein de liens où la gestion de mémoire est automatique.

2.4.7 Invocation de GSS_Release_buffer

Entrées :
- CHAINE D'OCTETS buffer.

Sorties :
- ENTIER major_status,
- ENTIER minor_status.

Codes d'état majeur en retour :
- GSS_S_COMPLETE indique que la mémorisation associée à la mémoire tampon d'entrée a bien été libérée.

- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Permet à l'appelant de libérer la mémorisation associée à une mémoire tampon de CHAINE D'OCTETS allouée par un autre appel GSS-API. Le comportement spécifique de cet appel dépend de l'environnement de langage et de programmation au sein duquel fonctionne une mise en œuvre GSS-API, et est donc détaillé au sein des spécifications de liens applicables ; en particulier, la mise en œuvre et l'invocation de cet appel peuvent être superflues (et peuvent être omises) au sein de liens où la gestion de mémoire est automatique.

2.4.8 Invocation de GSS_Release_OID_set

Entrées :
- o ENSEMBLE D'IDENTIFIANTS D'OBJET buffer.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status.

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que la mémorisation associée à l'ensemble d'identifiants d'objet entré a été bien libérée.
- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Permet aux appelants de libérer la mémorisation associée à l'objet Identifiant d'objet alloué par un autre appel GSS-API. Ce comportement spécifique d'appel dépend de l'environnement de langage et de programmation au sein duquel fonctionne une mise en œuvre GSS-API, et est donc détaillé au sein des spécifications de liens applicables ; en particulier, la mise en œuvre et l'invocation de cet appel peuvent être superflues (et peuvent être omises) au sein de liens où la gestion de mémoire est automatique.

2.4.9 Invocation de GSS_Create_empty_OID_set

Entrées :
- o (aucune)

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o ENSEMBLE D'IDENTIFIANTS D'OBJET oid_set -- l'appelant doit libérer avec GSS_Release_oid_set().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique la réussite de l'achèvement.
- o GSS_S_FAILURE indique que l'opération a échoué.

Crée un ensemble d'identifiant d'objet qui ne contient aucun identifiant d'objet, auquel les membres peuvent être ensuite ajoutés en utilisant le sous-programme GSS_Add_OID_set_member(). Ces sous-programmes sont destinés à être utilisés pour construire des ensembles d'identifiants d'objet de mécanisme, pour des entrées à GSS_Acquire_cred().

2.4.10 Invocation de GSS_Add_OID_set_member

Entrées :
- o IDENTIFIANT D'OBJET member_oid,
- o ENSEMBLE D'IDENTIFIANTS D'OBJET oid_set .

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique le bon achèvement.
- o GSS_S_FAILURE indique que l'opération a échoué.

Ajoute un identifiant d'objet à un ensemble d'identifiants d'objet. Ce sous-programme est destiné à être utilisé en

conjonction avec GSS_Create_empty_OID_set() lors de la construction d'un ensemble d'OID de mécanismes à entrer dans GSS_Acquire_cred().

2.4.11 Invocation de GSS_Test_OID_set_member

Entrées :
- o IDENTIFIANT D'OBJET member,
- o ENSEMBLE D'IDENTIFIANTS D'OBJET set.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o BOOLÉEN present.

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique la réussite de l'achèvement.
- o GSS_S_FAILURE indique que l'opération a échoué.

Interroge un ensemble d'identifiants d'objet pour déterminer si un identifiant d'objet spécifié est membre. Ce sous-programme est destiné à être utilisé avec les ensembles d'OID retournés par GSS_Indicate_mechs(), GSS_Acquire_cred(), et GSS_Inquire_cred().

2.4.12 Invocation de GSS_Inquire_names_for_mech

Entrée :
- o IDENTIFIANT D'OBJET input_mech_type, -- type de mécanisme.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o ENSEMBLE D'IDENTIFIANTS D'OBJET name_type_set -- l'appelant doit libérer avec GSS_Release_oid_set().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que le résultat name_type_set contient une liste de types de noms qui sont pris en charge par le mécanisme disponible en local identifié par input_mech_type.
- o GSS_S_BAD_MECH indique que le mécanisme identifié par input_mech_type n'était pas accepté au sein de la mise en œuvre locale, causant l'échec de l'interrogation.
- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Permet aux appelants de déterminer l'ensemble de types de noms qui sont acceptés par un mécanisme spécifique disponible en local.

2.4.13 Invocation de GSS_Inquire_mechs_for_name

Entrées :
- o NOM INTERNE input_name.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o ENSEMBLE D'IDENTIFIANTS D'OBJET mech_types -- l'appelant doit libérer avec GSS_Release_oid_set().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique qu'un ensemble d'identifiants d'objet, correspondant à l'ensemble des mécanismes qui conviennent au traitement de input_name, est disponible dans mech_types.
- o GSS_S_BAD_NAME indique que le input_name était mal formé et n'a pas pu être traité.
- o GSS_S_BAD_NAMETYPE indique que le paramètre input_name contenait un type de nom invalide ou un type de nom non accepté par la mise en œuvre GSS-API.
- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au

niveau de GSS-API.

Ce sous-programme retourne l'ensemble de mécanismes avec lesquels input_name peut être traité.

Chaque mécanisme retourné va reconnaître au moins un élément au sein du nom. Il est permis à ce sous-programme d'être mis en œuvre au sein d'une couche GSS-API indépendante du mécanisme, en utilisant les informations de type contenues dans le nom présenté, et fondé sur les information d'enregistrement fournies par les mises en œuvre individuelles de mécanisme. Cela signifie que les mech_types résultants retournés peuvent indiquer qu'un mécanisme particulier va comprendre un nom particulier lorsque en fait il refuserait d'accepter ce nom comme entrée à GSS_Canonicalize_name(), GSS_Init_sec_context(), GSS_Acquire_cred(), ou GSS_Add_cred() à cause de certaines propriétés du nom en question plutôt qu'une propriété du type de nom. Donc, ce sous-programme ne devrait être utilisé que comme pré-filtre pour un appel à un sous-programme spécifique de mécanisme ultérieur.

2.4.14 Invocation de GSS_Canonicalize_name

Entrées :
- o NOM INTERNE input_name,
- o IDENTIFIANT D'OBJET mech_type -- doit être un mécanisme explicite, pas un spécificateur ou identifiant "par défaut" de mécanisme de négociation.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o NOM INTERNE output_name. -- l'appelant doit libérer avec GSS_Release_name().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique qu'une réduction spécifique du mécanisme de input_name, telle que traitée par le mécanisme identifié par le mech_type, est disponible dans output_name.
- o GSS_S_BAD_MECH indique que le mécanisme identifié n'est pas accepté pour cette opération ; cela peut correspondre à un mécanisme qui n'est pas du tout accepté par la mise en œuvre GSS-API locale ou à un mécanisme de négociation avec lequel l'opération de canonisation ne peut pas être effectuée.
- o GSS_S_BAD_NAMETYPE indique que le nom entré ne contient pas un élément avec le type convenable pour être traité par le mécanisme identifié.
- o GSS_S_BAD_NAME indique que le nom entré contient un élément avec un type convenable pour être traité par le mécanisme identifié, mais que cet élément n'a pas pu être traité convenablement.
- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Ce sous-programme réduit un nom interne GSS-API input_name, qui peut en général contenir des éléments correspondants à plusieurs mécanismes, à un nom de mécanisme (MN) spécifique de mécanisme output_name en appliquant les traductions correspondant au mécanisme identifié par mech_type. Le contenu de input_name n'est pas affecté par l'opération GSS_Canonicalize_name(). Les références à output_name vont rester valides jusqu'à ce que output_name soit libéré, indépendamment du fait que input_name soit ou non ensuite libéré.

2.4.15 Invocation de GSS_Export_name

Entrées :
- o NOM INTERNE input_name, -- doit obligatoirement être un MN.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o CHAINE D'OCTETS output_name -- l'appelant doit libérer avec GSS_Release_buffer().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique qu'une représentation plate du nom entré est disponible dans output_name.
- o GSS_S_NAME_NOT_MN indique que le nom entré contenait des éléments correspondants à plusieurs mécanismes, de sorte qu'il ne peut pas être exporté dans une forme plate d'un seul mécanisme.
- o GSS_S_BAD_NAME indique que le nom entré était un MN, mais n'a pas pu être traité.
- o GSS_S_BAD_NAMETYPE indique que le nom entré était un MN, mais que son type n'est pas accepté par la mise en œuvre GSS-API.
- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au

niveau de GSS-API.

Ce sous-programme crée une représentation de nom plate, convenable pour la comparaison au bit près ou pour l'entrée dans GSS_Import_name() en conjonction avec l'OID d'objet Nom exporté GSS-API réservé, à partir d'un nom de mécanisme (MN de forme interne comme émis, par exemple, par GSS_Canonicalize_name() ou GSS_Accept_sec_context().

L'objet Nom exporté GSS-API émis est auto-descriptif ; aucun OID de niveau paramètre associé n'a besoin d'être émis par cet appel. Cette représentation plate consiste en une couche d'enveloppe indépendante du mécanisme, définie au paragraphe 3.2 de ce document, incluant une représentation de nom définie par le mécanisme.

Dans tous les cas, la sortie de nom plat par GSS_Export_name() pour correspondre à une entrée de MN particulière doit être invariable dans le temps au sein d'une installation particulière.

Le code d'état GSS_S_NAME_NOT_MN est fourni pour permettre aux mises en œuvre de rejeter les entrées de noms qui ne sont pas des MN. Il n'est cependant pas exigé pour les besoins de la conformité à la présente spécification que toutes les entrées de noms non MN soient nécessairement rejetées.

2.4.16 Invocation de GSS_Duplicate_name

Entrées :
- o NOM INTERNE src_name.

Sorties :
- o ENTIER major_status,
- o ENTIER minor_status,
- o NOM INTERNE dest_name. -- l'appelant doit libérer avec GSS_Release_name().

Codes d'état majeur en retour :
- o GSS_S_COMPLETE indique que dest_name fait référence à un objet Nom interne qui contient le même nom que celui passé à src_name.
- o GSS_S_BAD_NAME indique que l'entrée de nom était invalide.
- o GSS_S_FAILURE indique que l'opération demandée n'a pas pu être effectuée pour des raisons non spécifiées au niveau de GSS-API.

Ce sous-programme prend en entrée le nom interne src_name, et retourne une autre référence (dest_name) à ce nom qui peut être utilisée même si src_name est ultérieurement libéré. (Noter que ceci peut être mis en œuvre par copie ou par l'utilisation de comptes de référence.)

3. Définitions des structures de données pour l'usage de GSS-v2

Les paragraphes de cette section définissent, pour les besoins de l'interopérabilité et de la portabilité, certaines structures de données à utiliser avec GSS-v2.

3.1 Format de jeton indépendant du mécanisme

Ce paragraphe spécifie une représentation de niveau d'encapsulation indépendante du mécanisme pour le jeton initial d'une séquence GSS-API d'établissement de contexte, incorporant un identifiant du type de mécanisme à utiliser sur ce contexte et permettant aux jetons d'être interprétés sans ambiguïté chez les homologues GSS-API. L'utilisation de ce format est exigée pour les jetons initiaux d'établissement de contexte des mécanismes GSS-API en cours de normalisation dans l'Internet ; l'utilisation dans les jetons non initiaux est facultative.

Le format de codage pour l'étiquette de jeton déduite de l'ASN.1 et du DER (selon la syntaxe ASN.1 illustrative incluse plus loin dans ce paragraphe) mais sa représentation concrète est définie directement en termes d'octets plutôt qu'au niveau ASN.1 afin de faciliter les mises en œuvre interopérables sans utiliser le code de traitement ASN.1 général. L'étiquette de jeton comporte les éléments suivants, dans l'ordre :

1. 0x60 -- Étiquette pour [APPLICATION 0] SEQUENCE ; indique une forme construite, le codage de longueur défini suit.
2. Octets de longueur de jeton, spécifie la longueur des données qui suivent (c'est-à-dire, la somme des longueurs des

éléments 3 à 5 de cette liste, et de l'objet Jeton défini par le mécanisme qui suit l'étiquette). Cet élément comporte un nombre d'octets variable :

2a. Si la valeur indiquée est inférieure à 128, elle devra être représentée dans un seul octet avec le bit 8 (de poids fort) réglé à "0" et les bits restants représentant la valeur.

2b. Si la valeur indiquée fait 128 ou plus, elle devra être représentée sur deux octets ou plus, avec le bit 8 du premier octet à "1" et les bits restants du premier octet spécifiant le nombre d'octets supplémentaires. Les octets suivants portent la valeur, 8 bits par octet, chiffre de poids fort en premier. Le nombre minimum d'octets devra être utilisé pour coder la longueur (c'est-à-dire, aucun octet représentant des zéros en tête ne devra être inclus dans le codage de longueur).

3. 0x06 – Étiquette pour IDENTIFIANT D'OBJET
4. Longueur d'identifiant d'objet – longueur (nombre d'octets) de l'identifiant d'objet codé contenu dans un élément 5, codé selon les règles décrites en 2a. et 2b. ci-dessus.
5. Octets d'identifiant d'objet – nombre variable d'octets, codés selon les règles ASN.1 BER :

5a. Le premier octet contient la somme de deux valeurs : (1) le composant d'identifiant d'objet de niveau supérieur, multiplié par 40 (décimal) et (2) le composant d'identifiant d'objet de second niveau. Ce cas particulier est le seul point au sein d'un codage d'identifiant d'objet où un seul octet représente le contenu de plus d'un composant.

5b. Les octets suivants, si nécessaire, codent successivement les composants inférieurs dans l'identifiant d'objet représenté. Le codage d'un composant peut s'étendre sur plusieurs octets, en codant 7 bits par octet (le bit de poids fort en premier) et avec le bit 8 réglé à "1" sur tous les octets sauf l'octet final dans le codage du composant. Le nombre minimum d'octets devra être utilisé pour coder chaque composant (c'est-à-dire, aucun octet représentant des zéros en tête ne devra être inclus dans le codage d'un composant).

(Note : Dans de nombreuses mises en œuvre, les éléments 3 à 5 peuvent être mémorisés et référencés comme une chaîne contiguë constante.)

L'étiquette jeton est immédiatement suivie par un objet Jeton défini par le mécanisme. Noter qu'aucun spécificateur de taille indépendant n'intervient à la suite de la valeur d'identifiant d'objet pour indiquer la taille de l'objet Jeton défini par le mécanisme. Bien que l'usage de l'ASN.1 au sein des jetons définis par le mécanisme soit permis, il n'est pas exigé que les éléments de données spécifiques du mécanisme innerContextToken, innerMsgToken, et sealedUserData emploient les conventions de codage ASN.1 BER/DER.

La syntaxe ASN.1 suivante est incluse pour les seuls besoins de la descriptive pour illustrer les relations structurelles entre les objets jeton et étiquette. Pour les besoins de l'interopérabilité, le codage des jetons et étiquettes devra être effectué en utilisant les procédures concrètes de codage décrites plus haut dans ce paragraphe.

```
DEFINITIONS GSS-API ::=

DÉBUT

MechType ::= IDENTIFIANT D'OBJET
        -- Les appelants de définitions de structure de données doivent être capables de distinguer entre les éléments
           de données InitialContextToken, SubsequentContextToken, PerMsgToken, et SealedMessage sur la base de
           l'usage dans lequel ils surviennent.

InitialContextToken ::=     [APPLICATION 0] SEQUENCE IMPLICITE {
        -- indication d'option (délégation, etc.) au sein du jeton spécifique du mécanisme.
    thisMech MechType,
    innerContextToken TOUT DÉFINI PAR thisMech
        -- contenu spécifique du mécanisme ; structure ASN.1 non exigée.
    }

SubsequentContextToken ::= innerContextToken TOUT
        -- interprétation fondée sur InitialContextToken précédent ; structure ASN.1 non exigée.

PerMsgToken ::=
        -- comme émis par GSS_GetMIC et traité par GSS_VerifyMIC ; structure ASN.1 non exigée.
    innerMsgToken TOUT

SealedMessage ::=
        -- comme émis par GSS_Wrap et traité par GSS_Unwrap ; inclut un indicateur interne, défini par le
           mécanisme, du chiffrement ou non chiffrement ; structure ASN.1 non exigée.
    sealedUserData TOUT
FIN
```

3.2 Format d'objet Nom exporté indépendant du mécanisme

Ce paragraphe spécifie un niveau indépendant du mécanisme de représentation d'encapsulation pour les noms exportés via l'appel GSS_Export_name() incluant un identifiant d'objet représentant le mécanisme exportateur. Le format des noms encapsulés via cette représentation devra être défini dans les documents de mécanisme individuels. La valeur de l'identifiant d'objet pour indiquer les noms de ce type est définie au paragraphe 4.7.

Aucun OID de type de nom n'est inclus dans ce niveau indépendant du mécanisme de définition de format, car (selon les spécifications de mécanisme individuel) le nom inclus peut être typé implicitement ou explicitement en utilisant un moyen autre que le codage d'un OID.

Les octets dans les éléments MECH_OID_LEN et NAME_LEN sont représentés avec l'octet de poids fort en premier (équivalent à l'ordre des octets de réseau dans IP).

Longueur	Nom	Description
2	TOK_ID	Identifiant de jeton. Pour les objets Nom exporté, doit être hex 04 01.
2	MECH_OID_LEN	Longueur de l'OID de mécanisme.
	MECH_OID_LEN	
	MECH_OID	OID de mécanisme, en DER.
4	NAME_LEN	Longueur du nom.
	NAME_LEN	
	NAME	Nom exporté ; format défini dans la RFC de mécanisme applicable.

Un exemple concret du contenu d'un objet Nom exporté, déduit du mécanisme Kerberos version 5, est le suivant :

04 01 00 0B 06 09 2A 86 48 86 F7 12 01 02 02 hx xx xx xl pp qq ... zz

04 01 identifiant de jeton obligatoire
00 0B longueur sur deux octets de la valeur ASN.1 codée en DER de l'OID de type qui suit immédiatement, octet de poids fort en premier.
06 09 2A 86 48 86 F7 12 01 02 02 valeur ASN.1 codée en DER de l'OID de type ; l'OID de mécanisme Kerberos v5 indique le nom exporté Kerberos v5.
en détail : 06 octet identifiant (6 = OID)
09 octet de longueur
2A 86 48 86 F7 12 01 02 02 octets de contenu
hx xx xx xl longueur sur 4 octets du nom blob exporté qui suit immédiatement, octet de poids fort en premier
pp qq ... zz nom blob exporté de la longueur spécifiée, bits et octets spécifiés dans la spécification de mécanisme GSS-API v2 (Kerberos 5).

4. Définitions des types de noms

Cette section comporte les définitions des types de nom et des syntaxes associées qui sont définies de façon indépendante du mécanisme au niveau GSS-API plutôt que d'être définies dans les spécifications de mécanisme individuel.

4.1 Forme de nom de service fondé sur l'hôte

Cette forme de nom devra être représentée par l'identifiant d'objet :
{iso(1) member-body(2) United States(840) mit(113554) infosys(1) "gssapi(2) generic(1) service_name(4)}.

Le nom symbolique recommandé pour ce type est "GSS_C_NT_HOSTBASED_SERVICE".

Pour des raisons de compatibilité avec les mises en œuvre existantes, il est recommandé que cet OID soit utilisé plutôt que l'autre valeur incluse dans la [RFC2078] :
{1(iso), 3(org), 6(dod), 1(internet), 5(security), 6(nametypes), 2(gss-host-based-services)}

Bien qu'il ne soit pas recommandé que cette autre valeur soit émise en sortie par les mises en œuvre GSS, il est recommandé qu'elle soit acceptée en entrée comme équivalente à la valeur recommandée.

Ce type de nom est utilisé pour représenter des services associés aux ordinateurs hôtes. La prise en charge de cette forme de nom est recommandée au concepteurs de mécanisme dans l'intérêt de la portabilité, mais n'est pas rendue obligatoire par la présente spécification. Cette forme de nom est construite en utilisant deux éléments, "service" et "nom_d'hôte", comme suit : service@nom_d'hôte

Lorsque une référence à un nom de ce type est résolue, le "nom_d'hôte" peut (comme exemple de stratégie de mise en œuvre) être canonisée en tentant une recherche DNS et en utilisant le nom de domaine pleinement qualifié qui est retourné, ou en utilisant le "nom_d'hôte" comme fourni si la recherche DNS échoue. L'opération de canonisation transpose aussi le nom de l'hôte en caractères minuscules.

L'élément "nom_d'hôte" peut être omis. Si un séparateur "@" n'est pas inclus, le nom entier est interprété comme étant le spécificateur du service, avec le "nom_d'hôte" pris par défaut comme le nom canonisé de l'hôte local.

Les documents qui spécifient les moyens de l'intégration de GSS dans un protocole particulier devraient déclarer que :
(a) un nom spécifique enregistré par l'IANA associé à ce protocole devra être utilisé pour l'élément "service" (cela admet, si nécessaire, la possibilité qu'un seul nom soit enregistré et partagé dans un ensemble de protocoles en rapport), ou
(b) que le nom générique "hôte" devra être utilisé pour l'élément "service", ou
(c) pour ce protocole, des positions de repli dans l'ordre (a, puis b) ou (b, puis a) devront être appliquées.

L'enregistrement par l'IANA de noms spécifiques selon (a) devrait être traité conformément à la politique d'allocation "spécification exigée", définie par le BCP 26, RFC2434 comme suit : "Les valeurs et leur signification doivent être documentées dans une RFC ou autre référence disponible, en détail suffisant pour que soit possible l'interopérabilité entre les mises en œuvre indépendantes."

4.2 Forme de nom d'usager

Cette forme de nom devra être représentée par l'identifiant d'objet {iso(1) member-body(2) United States(840) mit(113554) infosys(1) gssapi(2) generic(1) user_name(1)}. Le nom symbolique, indépendant du mécanisme, recommandé pour ce type, est "GSS_C_NT_USER_NAME". (Noter que la même forme de nom et d'OID est définie dans le mécanisme GSS-API Kerberos v5, mais le nom symbolique qui y est recommandé commence par un préfixe "GSS_KRB5_NT_".)

Ce type de nom est utilisé pour indiquer un usager désigné sur un système local. Sa syntaxe et son interprétation peuvent être spécifiques du système d'exploitation. Cette forme de nom est construite comme : nom_d'usager

4.3 Forme d'UID de machine

Cette forme de nom devra être représentée par l'identifiant d'objet {iso(1) member-body(2) United States(840) mit(113554) infosys(1) gssapi(2) generic(1) machine_uid_name(2)}. Le nom symbolique, indépendant du mécanisme, recommandé pour ce type, est "GSS_C_NT_MACHINE_UID_NAME". (Noter que la même forme de nom et d'OID est définie dans le mécanisme GSS-API Kerberos v5, mais le nom symbolique qui y est recommandé commence par un préfixe "GSS_KRB5_NT_".)

Ce type de nom est utilisé pour indiquer identifiant d'utilisateur numérique correspondant à un usager sur un système local. Son interprétation est spécifique du système d'exploitation. La gss_buffer_desc qui représente un nom de ce type devrait contenir un identifiant d'utilisateur à signification locale, représenté dans l'ordre des octets de l'hôte. L'opération GSS_Import_name() résout cet uid en un nom_d'usager, qui est alors traité comme forme Nom_d'usager.

4.4 Forme d'UID de chaîne

Cette forme de nom devra être représentée par l'identifiant d'objet {iso(1) member-body(2) United States(840) mit(113554) infosys(1) gssapi(2) generic(1) string_uid_name(3)}. Le nom symbolique recommandé pour ce type, est "GSS_C_NT_STRING_UID_NAME". (Noter que la même forme de nom et d'OID est définie dans le mécanisme GSS-API Kerberos v5, mais le nom symbolique qui y est recommandé commence par un préfixe "GSS_KRB5_NT_".)

Ce type de nom est utilisé pour indiquer une chaîne de chiffres représentant l'identifiant d'utilisateur numérique d'un usager sur un système local. Son interprétation est spécifique du système d'exploitation. Ce type de nom est similaire à la forme d'UID de machine, sauf que la mémoire tampon contient une chaîne qui représente l'identifiant d'utilisateur.

4.5 Type de nom anonyme

La valeur d'identifiant d'objet suivante est fournie comme moyen d'identifier les noms "anonymes", et peut leur être comparée afin de déterminer, d'une façon indépendante du mécanisme, si un nom se réfère à un principal anonyme : {1(iso), 3(org), 6(dod), 1(internet), 5(security), 6(nametypes), 3(gss-anonymous-name)}

Le nom symbolique recommandé correspondant à cette définition est GSS_C_NT_ANONYMOUS.

4.6 GSS_C_NO_OID

Le nom symbolique recommandé GSS_C_NO_OID correspond à une valeur d'entrée nulle au lieu d'un identifiant d'objet réel. Lorsque il est spécifié, il indique l'interprétation d'un nom associé sur la base d'une syntaxe imprimable par défaut spécifique du mécanisme.

4.7 Objet de nom exporté

Les objets de nom du type d'objet Nom exporté indépendant du mécanisme, comme définis au paragraphe 3.2, seront identifiés par l'identifiant d'objet suivant : {1(iso), 3(org), 6(dod), 1(internet), 5(security), 6(nametypes), 4(gss-api-exported-name)}

Le nom symbolique recommandé correspondant à cette définition est GSS_C_NT_EXPORT_NAME.

4.8 GSS_C_NO_NAME

Le nom symbolique recommandé GSS_C_NO_NAME indique qu'aucun nom n'est passé dans une valeur particulière d'un paramètre utilisé pour les besoins de transfert de noms. Noter que GSS_C_NO_NAME n'est pas un type de nom réel, et n'est pas représenté par un OID ; son acceptabilité au lieu d'un nom réel est restreinte aux appels spécifiques (GSS_Acquire_cred(), GSS_Add_cred(), et GSS_Init_sec_context()) des usages identifiés dans la présente spécification.

5. Exemple de scénarios spécifiques de mécanisme

Cette section donne des exemples qui illustrent l'utilisation de divers types de mécanismes candidats à la prise en charge de GSS-API. Ces discussions sont principalement destinées aux lecteurs familiarisés avec les technologies spécifiques de la sécurité, montrant comment les fonctions GSS-API peuvent être utilisées et mises en œuvre par les mécanismes candidats sous-jacents. Ils ne devraient pas être regardés comme des contraintes pour les mises en œuvre ou comme définissant le seul moyen par lequel les fonctions GSS-API peuvent être réalisées avec une technologie sous-jacente particulière, et ne montrent pas toutes les caractéristiques de GSS-API avec chaque technologie.

5.1 Kerberos v5, un seul TGT

Les fonctions de connexion spécifiques d'un système d'exploitation donnent un ticket distributeur de tickets (TGT, *ticket-granting ticket*) au serveur Kerberos de domaine local ; le TGT est placé dans une structure d'accréditifs pour le client. Le client invoque GSS_Acquire_cred() pour acquérir un cred_handle afin de référencer les accréditifs à utiliser dans l'établissement des contextes de sécurité.

Le client invoque GSS_Init_sec_context(). Si le service demandé est situé dans un domaine différent, GSS_Init_sec_context() obtient les paires de TGT/clé nécessaires pour traverser le chemin du domaine local au domaine cible ; ces données sont placées dans l'antémémoire de TGT du propriétaire. Après toute résolution de domaine distant nécessaire, GSS_Init_sec_context() donne un ticket de service au service demandé avec une clé de session correspondante ; ces données sont mémorisées en conjonction avec le contexte. Le code GSS-API envoie la ou les demandes KRB_TGS_REQ et reçoit la ou les réponses KRB_TGS_REP (dans le cas de réussite) ou KRB_ERROR.

En supposant la réussite, GSS_Init_sec_context() construit un message KRB_AP_REQ au format Kerberos, et le retourne dans un jeton de résultat output_token. Le client envoie le output_token au service.

Le service passe le jeton reçu comme argument du jeton d'entrée à GSS_Accept_sec_context() qui vérifie l'authentifiant,

fournit au service le nom authentifié du client, et retourne un output_context_handle *(lien de contexte de sortie)*.

Les deux parties détiennent maintenant la clé de session associée au ticket de service, et peuvent utiliser cette clé dans les opérations GSS_GetMIC(), GSS_VerifyMIC(), GSS_Wrap(), et GSS_Unwrap() suivantes.

5.2 Kerberos V5, double TGT

L'acquisition du TGT est comme ci-dessus.

Note : Pour éviter des invocations inutilement fréquentes de chemins erronés lors de la mise en œuvre de GSS-API par dessus Kerberos v5, il semble approprié de représenter "K-V5 à un seul TGT" et "K-V5 à double TGT" avec des mech_types distincts, et cet exposé retient cette hypothèse.

Sur la base du type de mécanisme (spécifié ou par défaut) GSS_Init_sec_context() détermine que le protocole double TGT devrait être employé pour la cible spécifiée. GSS_Init_sec_context() retourne l'état majeur GSS_S_CONTINUE_NEEDED, et son jeton de sortie retourné contient une demande au service du TGT du service. (Si il existe déjà en antémémoire un TGT de service avec une durée de vie restante suffisamment longue, il peut être utilisable, supprimant le besoin de cette étape.) Le client passe le jeton de sortie au service. Noter que ce scénario illustre une utilisation différente de la facilité de retour de l'état GSS_S_CONTINUE_NEEDED que pour la prise en charge de l'authentification mutuelle ; noter que les deux utilisations peuvent coexister comme opérations successives au sein d'une seule opération d'établissement de contexte.

Le service passe le jeton reçu comme argument du jeton d'entrée à GSS_Accept_sec_context(), qui le reconnaît comme une demande de TGT. (Noter que Kerberos v5 actuel ne définit pas de mécanisme intra protocole pour représenter une telle demande.) GSS_Accept_sec_context() retourne l'état majeur GSS_S_CONTINUE_NEEDED et fournit le TGT du service dans son jeton de sortie. Le service envoie le jeton de sortie au client.

Le client passe le jeton reçu comme argument du jeton d'entrée à une continuation de GSS_Init_sec_context(). GSS_Init_sec_context() met en antémémoire le TGT de service reçu et l'utilise au titre d'une demande de ticket de service au serveur d'authentification Kerberos, mémorisant le ticket de service retourné et la clé de session en conjonction avec le contexte. GSS_Init_sec_context() construit un authentifiant au format Kerberos, et le retourne dans le jeton de sortie avec le retour d'état majeur GSS_S_COMPLETE. Le client envoie le jeton de sortie au service.

Le service passe le jeton reçu comme argument du jeton d'entrée à un appel de continuation à GSS_Accept_sec_context(). GSS_Accept_sec_context() vérifie l'authentifiant, fournit au service le nom authentifié du client, et retourne l'état majeur GSS_S_COMPLETE.

GSS_GetMIC(), GSS_VerifyMIC(), GSS_Wrap(), et GSS_Unwrap() sont comme ci-dessus.

5.3 Cadre d'authentification X.509

Cet exemple illustre l'utilisation de GSS-API en conjonction avec les mécanismes à clé publique, en cohérence avec le cadre d'authentification d'annuaire X.509.

L'appel GSS_Acquire_cred() établit une structure d'accréditifs, rendant la clé privée du client accessible pour être utilisée au nom du client.

Le client invoque GSS_Init_sec_context(), qui interrogé le répertoire pour acquérir (et valider) une chaîne de certificats de clé publique, collectant par là la clé publique du service. L'opération de validation de certificat détermine que les vérifications d'intégrité convenables ont été appliquées par des autorités de confiance et que ces certificats ne sont pas périmés. GSS_Init_sec_context() génère une clé secrète à utiliser dans les opérations de protection par message sur le contexte, et chiffre cette clé secrète sous la clé publique du service.

La clé secrète chiffrée, ainsi qu'une quantité d'authentifiant, signées avec la clé privée du client, est incluse dans le jeton de sortie provenant de GSS_Init_sec_context(). Le jeton de sortie porte aussi un chemin de certification, consistant en une chaîne de certificats conduisant du service au client ; une variante de cette approche différerait cette résolution de chemin pour qu'elle soit effectuée par le service plutôt que d'être affirmée par le client. L'application du client envoie le jeton de sortie au service.

Le service passe le jeton reçu comme argument du jeton d'entrée à GSS_Accept_sec_context(). GSS_Accept_sec_context()

valide le chemin de certification, et détermine en sortie un lien certifié entre le nom distinctif du client et la clé publique du client. Avec cette clé publique, GSS_Accept_sec_context() peut traiter la quantité authentifiante du jeton d'entrée et vérifier que la clé privée du client a été utilisée pour signer le jeton d'entrée. À ce moment, le client est authentifié auprès du service. Le service utilise sa clé privée pour déchiffrer la clé secrète chiffrée qui lui a été fournie pour les opérations de protection par message sur le contexte.

Le client invoque GSS_GetMIC() ou GSS_Wrap() sur un message de données, ce qui cause l'application des facilités d'authentification, d'intégrité, et (facultativement) de confidentialité par message à ce message. Le service utilise la clé secrète partagée du contexte pour effectuer les invocations GSS_VerifyMIC() et GSS_Unwrap() correspondantes.

6. Considérations pour la sécurité

Le présent document spécifie une interface de service pour les facilités et services de sécurité ; à ce titre, les considérations de sécurité sont examinées tout au long de la spécification. Néanmoins, il est approprié de résumer certains points spécifiques pertinents pour les mises en œuvre de GSS-API et pour les applications appelantes. L'usage de l'interface GSS-API ne fournit pas par lui-même de service ou assurance de sécurité ; ces attributs sont plutôt dépendants du ou des mécanismes sous-jacents qui prennent en charge une mise en œuvre de GSS-API. Les appelants doivent être attentifs aux demandes faites aux invocations de GSS-API et aux indicateurs d'état retournés par GSS-API, car ils spécifient les caractéristiques du service de sécurité que va fournir GSS-API. Lorsque la facilité de transfert de contexte interprocessus est utilisée, les contrôles locaux appropriés devraient être appliqués pour restreindre l'accès aux jetons interprocès et aux données sensibles qu'ils contiennent.

7. Activités connexes

Afin de mettre en œuvre la GSS-API par dessus les mécanismes de sécurité existants, émergeants, et futurs :
- les identifiants d'objet doivent être alloués aux mécanismes GSS-API candidats et aux types de noms qu'ils acceptent,
- les formats concrets d'élément de données et les procédures de traitement doivent être définies pour les mécanismes candidats,
- les applications appelantes doivent mettre en œuvre les conventions de formatage qui vont leur permettre de distinguer les jetons GSS-API des autres données portées dans leur protocoles d'application,
- des liens de langage concret sont nécessaires pour les environnements de programmation dans lesquels la GSS-API sera employée, comme la [RFC1509] en définit pour le langage de programmation C et GSS-v1. Les liens de langage C pour GSS-v2 sont définis dans la [RFC2744].

8. Documents de référence

[ISO-7498-2] International Standard ISO 7498-2-1988(E), "Security Architecture".

[ISOIEC-8824] ISO/IEC 8824, "Specification of Abstract Syntax Notation One (ASN.1)".

[ISOIEC-8825] ISO/IEC 8825, "Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)".)

[RFC1507] C. Kaufman, "DASS : Service de sécurité à authentification répartie", septembre1993. *(Expérimentale)*

[RFC1508] J. Linn, "Interface générique de programme de service de sécurité", septembre 1993. *(P.S.) (remplacée par* RFC2078, *elle-même remplacée par* RFC2743*)*

[RFC1509] J. Wray, "API de service générique de sécurité : liens C", septembre 1993. *(remplacée par* RFC2744*)*

[RFC1964] J. Linn, "Mécanisme GSS-API de Kerberos version 5", juin 1996. (*MàJ par* RFC4121 *et* RFC6649)

[RFC2025] C. Adams, "Mécanisme simple de GSS-API à clé publique (SPKM)", octobre 1996. *(P.S.)*

[RFC2078] J. Linn, "Interface générique de programme d'application de service de sécurité, version 2", janvier 1997. *(Rendue obsolète par la présente RFC.)*

[RFC2203] M. Eisler, A. Chiu, L. Ling, "Spécification du protocole RPCSEC_GSS", septembre 1997. *(P.S., MàJ par* RFC5403)

[RFC2744] J. Wray, "API de service générique de sécurité, version 2 : liaisons C", janvier 2000. *(P.S.)*

Appendice A Contraintes pour la conception des mécanismes

Les contraintes suivantes sur les concepts de mécanisme GSS-API sont adoptées en réponse aux exigences observées du protocole d'appel, et il est prévu que les descriptions ultérieures de mécanisme GSS-API y adhéreront pour les documents en cours de normalisations des spécifications de l'Internet.

Il est vivement recommandé que les mécanismes qui offrent des services de protection par message offrent aussi au moins un des services de détection de répétition et de séquençage, car les mécanismes qui n'offrent ni l'un ni l'autre ne satisferont pas aux exigences de certains des protocoles d'appel candidats.

Appendice B Compatibilité avec GSS-v1

Il est dans les intentions du présent document de définir une interface et des procédures qui préservent la compatibilité entre les appelants GSS-v1 [RFC1508] et les fournisseurs GSS-v2. Tous les appels définis dans GSS-v1 sont préservés, et il a été un des objectifs que les appelants GSS-v1 soient capables de fonctionner par dessus les mises en œuvre de fournisseur GSS-v2. Certains changements détaillés, résumés dans cette section, ont été fait afin de résoudre des omissions identifiées dans GSS-v1.

Les constructions GSS-v1 suivantes, bien que prise en charge au sein de GSS-v2, sont déconseillées.

Noms des sous-programmes de traitement par message : GSS_Seal() déconseillé en faveur de GSS_Wrap() ; GSS_Sign() d"conseillé en faveur de GSS_GetMIC() ; GSS_Unseal() déconseillé en faveur de GSS_Unwrap() ; GSS_Verify() déconseillé en faveur de GSS_VerifyMIC().

La facilité GSS_Delete_sec_context() pour l'utilisation du jeton de contexte, qui permet aux mécanismes de signaler la suppression de contexte, est conservé pour la compatibilité avec GSS-v1. Pour l'usage courant, il est recommandé que les deux homologues d'un contexte invoquent GSS_Delete_sec_context() de façon indépendante, passant une mémoire tampon nulle de output_context_token pour indiquer qu'aucun jeton de contexte n'est exigé. Les mises en œuvre de GSS_Delete_sec_context() devraient supprimer les informations pertinentes de contexte mémorisées en local.

La présent spécification GSS-v2 ajoute les appels suivants qui ne sont pas présents dans GSS-v1:

Appels de gestion d'accréditif : GSS_Add_cred(), GSS_Inquire_cred_by_mech().

Appels de niveau contexte : GSS_Inquire_context(), GSS_Wrap_size_limit(), GSS_Export_sec_context(), GSS_Import_sec_context().

Appels par message : Pas de nouveaux appels. Les appels existants ont été renommés.

Appels de soutien : GSS_Create_empty_OID_set(), GSS_Add_OID_set_member(), GSS_Test_OID_set_member(), GSS_Inquire_names_for_mech(), GSS_Inquire_mechs_for_name(), GSS_Canonicalize_name(), GSS_Export_name(), GSS_Duplicate_name().

La présent spécification GSS-v2 introduit trois nouvelles facilités applicables aux contextes de sécurité, indiquées en utilisant les valeurs d'état de contexte suivantes qui ne sont pas présentes dans GSS-v1 :

- anon_state, réglé à VRAI pour indiquer que l'initiateur d'un contexte est anonyme du point de vue de la cible ; le paragraphe 1.2.5 donne une description résumée de la facilité de prise en charge de l'anonymat par GSS-v2, dont la prise en charge et l'utilisation sont facultatives.
- prot_ready_state, réglé à VRAI pour indiquer qu'un contexte peut être utilisé pour la protection par message avant la fin de l'achèvement de l'établissement de contexte ; le paragraphe 1.2.7 donne une description résumée de la facilité GSS-v2 d'activer des mécanismes pour permettre de façon sélective une protection par message durant l'établissement de contexte, dont la prise en charge et l'utilisation sont facultatives.
- trans_state, réglé à VRAI pour indiquer qu'un contexte est transférable à un autre processus en utilisant la facilité GSS-v2 GSS_Export_sec_context().

Ces valeurs d'état sont représentées (au niveau des liens C) dans des positions au sein d'un vecteur binaire qui ne sont pas utilisées dans GSS-v1, et peuvent être ignorées en toute sécurité par les appelants GSS-v1.

De nouvelles entrées conf_req_flag et integ_req_flag sont définies pour GSS_Init_sec_context(), principalement pour fournir des informations aux mécanismes de négociation. Cela introduit un problème de compatibilité avec les appelants GSS-v1, discuté au paragraphe 2.2.1.

Par rapport à GSS-v1, GSS-v2 fournit des lignes directrices supplémentaires pour les mises en œuvre GSS-API dans les domaines suivants : la robustesse de mise en œuvre, la gestion des accréditifs, le comportement dans les configurations multi mécanismes, la prise en charge des dénominations, et l'inclusion des services facultatifs de séquençage. La facilité d'étiquetage de jeton comme défini au paragraphe 3.1 de GSS-v2, est maintenant décrite directement en termes d'octets pour faciliter l'interopérabilité de mise en œuvre sans code de traitement ASN.1 général ; la syntaxe ASN.1 correspondante, incluse à des fins descriptives, est inchangée par rapport à celle de GSS-v1. Pour être utilisée en conjonction avec les facilités de prise en charge de dénomination, une nouvelle construction d'objet Nom exporté est ajoutée. Des types de nom supplémentaires sont introduits à la Section 4.

La présente spécification GSS-v2 ajoute les valeurs d'état majeur suivantes qui ne sont pas définies dans GSS-v1 :

GSS_S_BAD_QOP	valeur de QOP non prise en charge
GSS_S_UNAUTHORIZED	opération non autorisée
GSS_S_UNAVAILABLE	opération indisponible
GSS_S_DUPLICATE_ELEMENT	l'élément d'accréditif demandé est dupliqué
GSS_S_NAME_NOT_MN	le nom contient des éléments multi mécanismes
GSS_S_GAP_TOKEN	détection d'un ou de jetons précédents sautés

De ces ajouts de codes d'état, seuls deux valeurs sont définies comme pouvant être retournées par les appels existant dans GSS-v1 : GSS_S_BAD_QOP (retournable par GSS_GetMIC() et GSS_Wrap()), et GSS_S_GAP_TOKEN (retournable par GSS_VerifyMIC() et GSS_Unwrap()).

De plus, les descriptions GSS-v2 de certains appels présents dans GSS-v1 ont été mises à jour pour permettre le retour de valeurs supplémentaires d'état majeur par rapport à l'ensemble défini dans GSS-v1 : GSS_Inquire_cred() a GSS_S_DEFECTIVE_CREDENTIAL et GSS_S_CREDENTIALS_EXPIRED définis comme retournables, GSS_Init_sec_context() a GSS_S_OLD_TOKEN, GSS_S_DUPLICATE_TOKEN, et GSS_S_BAD_MECH définis comme retournables, et GSS_Accept_sec_context() a GSS_S_BAD_MECH défini comme retournable.

Appendice C Changements par rapport à la RFC2078

Le présent document incorpore un certain nombre de changements par rapport à la RFC-2078, faits principalement en réponse à l'expérience de mise en œuvre, pour les besoins de l'alignement avec les liens de langage C de GSS-v2, et pour apporter des précisions de caractère informatif. Cette section résume les changements techniques incorporés.

Généralités :

- Précise l'usage des sous-programmes de libération d'objet, et incorpore la déclaration que certains peuvent être omis dans certains environnements de fonctionnement.
- Retrait des sous-programmes GSS_Release_OID, GSS_OID_to_str(), et GSS_Str_to_OID().
- Précise les circonstances dans lesquelles des jetons de longueur zéro peuvent exister valablement comme entrées et sorties d'appels GSS-API.
- Ajout du code d'état GSS_S_BAD_MIC comme autre nom pour GSS_S_BAD_SIG.
- Pour GSS_Display_status(), le choix de retourner plusieurs valeurs d'état en parallèle ou via itération est renvoyé aux liens de langage, et ajout de commentaires déconseillant le retour de GSS_S_CONTINUE_NEEDED.
- Précisions adaptées et incorporées sur la prise en charge de service facultatif, de la délégation, et du transfert de contexte interprocès provenant du document sur les liens C.
- Ajout et mise à jour de références aux documents en rapport, et au statut actuel de l'OID du mécanisme Kerberos.
- Ajout d'une déclaration générale sur les appels GSS-API qui n'ont pas d'effet collatéral visible au niveau GSS-API.

En rapport avec le contexte (y compris les questions de protection par message) :

- Précise l'usage de GSS_Delete_sec_context() pour les contextes partiellement établis.
- Ajout d'une précision sur le comportement et l'usage du contexte sur GSS_Export_sec_context() et GSS_Import_sec_context() à la suite d'une séquence d'exportation-importation.
- Ajout des entrées informatives conf_req_flag, integ_req_flag à GSS_Init_sec_context(). (Noter que cette facilité introduit une rétro incompatibilité avec les appelants GSS-v1, discutée au paragraphe 2.2.1 ; cette implication a été reconnue et acceptée dans les discussions du groupe de travail.)
- Déclaration que GSS_S_FAILURE est à retourner si GSS_Init_sec_context() ou GSS_Accept_sec_context() est passé comme lien d'un contexte qui est déjà pleinement établi.

- Sur GSS_Inquire_sec_context(), déclaration que le nom de source et le nom de cible ne sont pas retournés tant que l'état GSS_S_COMPLETE n'est pas atteint ; suppression du code d'état GSS_S_CONTEXT_EXPIRED (remplacé par la valeur EXPIRÉ de retour de durée de vie) ; déclaration de l'exigence de conserver les données interrogeables jusqu'à la libération du contexte par l'appelant ; ajout de la valeur de résultat indiquant si le contexte est ou non pleinement ouvert.
- Ajout de la discussion des conditions d'interopérabilité pour les mécanismes qui permettent la prise en charge facultative des QOP. Retrait des référence aux éléments structurés de QOP dans GSS_Verify_MIC().
- Ajout d'une discussion de l'utilisation de l'état GSS_S_DUPLICATE_TOKEN pour indiquer les jetons par message reflétés.
- Précision de l'utilisation des codes de séquençage informatif des appels de protection par message en conjonction avec les retours d'état majeur GSS_S_COMPLETE et GSS_S_FAILURE, ajustant en conséquence les descriptions de code d'état.
- Ajout de déclarations spécifiques sur l'impact des défaillances GSS_GetMIC() et GSS_Wrap() sur les informations d'état de contexte, et généralisation des déclarations existantes sur l'impact des défaillances de traitement sur les jetons par message reçus.
- Pour GSS_Init_sec_context() et GSS_Accept_sec_context(), permettre que le type de mécanisme retourné soit valide avant GSS_S_COMPLETE, reconnaissant que la valeur peut changer sur des appels de continuation successifs dans le cas de mécanisme négocié.
- Suppression de l'état GSS_S_CONTEXT_EXPIRED de GSS_Import_sec_context().
- Ajout de l'entrée conf_req_flag à GSS_Wrap_size_limit().
- Déclaration de l'exigence pour la prise en charge de mécanismes des services de protection par message comme étant utilisables concurremment dans les deux directions sur un contexte.

Au sujet des accréditifs :

- Pour GSS_Acquire_cred() et GSS_Add_cred(), alignement avec la déclaration de liens C de la non prise en charge probable des accréditifs INITIATE ou BOTH si le nom entré n'est ni vide ni un nom résultant de l'application de GSS_Inquire_cred() à l'accréditif par défaut. De plus, on déclare qu'un nom explicite retourné par GSS_Inquire_context() devrait aussi être accepté. Ajout d'un commentaire sur la possible variation dans le temps des résultats de la résolution par défaut et ses implications. Alignement avec le comportement des liens C lorsque GSS_C_NO_NAME est fourni pour le nom désiré. Dans GSS_Acquire_cred(), on déclare que NUL, plutôt que l'ensemble d'OID vide, devrait être utilisé pour les mécanismes désirés afin de demander l'ensemble de mécanismes par défaut.
- Ajout de GSS_S_CREDENTIALS_EXPIRED comme état majeur retournable pour GSS_Acquire_cred(), GSS_Add_cred(), spécifiant aussi GSS_S_NO_CRED comme retour approprié pour l'indisponibilité d'accréditif temporaire, fixable par l'utilisateur. GSS_Acquire_cred() et GSS_Add_cred() vont aussi retourner GSS_S_NO_CRED si une défaillance d'autorisation se rencontre lors de l'acquisition des accréditifs.
- Retrait du retour d'état GSS_S_CREDENTIALS_EXPIRED de la protection par message, des appels GSS_Context_time(), et GSS_Inquire_context().
- Pour GSS_Add_cred(), alignement de la description du comportement de liens C lorsque l'ajout d'éléments est demandé à l'accréditif par défaut.
- Mise à niveau de l'algorithme de résolution d'accréditif par défaut recommandé au statut d'exigence pour les accréditifs d'initiateur.
- Pour GSS_Release_cred(), GSS_Inquire_cred(), et GSS_Inquire_cred_by_mech(), précision du comportement pour l'entrée de GSS_C_NO_CREDENTIAL.

En rapport avec le nom :

- Alignement de la description de GSS_Inquire_mechs_for_name() sur les liens C.
- Retrait du retour d'état GSS_S_BAD_NAMETYPE de GSS_Duplicate_name(), GSS_Display_name() ; son applicabilité est restreinte à GSS_Compare_name().
- Alignement sur la déclaration des liens C du comportement GSS_Import_name() avec l'entrée de type de nom GSS_C_NO_OID, et mention que les spécifications de mécanisme GSS-v2 doivent définir des procédures de traitement applicables à leurs mécanismes. Précision aussi de l'usage de GSS_C_NO_OID avec GSS_Display_name().
- Dégradation de la référence à la canonisation de nom via une recherche DNS au rang d'exemple.
- Pour GSS_Canonicalize_name(), mention que ni les mécanismes négociés ni le mécanisme par défaut n'acceptent en entrée les mech_types pour cette opération, et spécifie que l'état GSS_S_BAD_MECH est retourné dans ce cas. Précise que l'opération GSS_Canonicalize_name() est non destructive pour son nom d'entrée.
- Précisé la sémantique du type de nom GSS_C_NT_USER_NAME.
- Ajout des descriptions de types de nom supplémentaires. Ajout aussi de la discussion de GSS_C_NO_NAME et des restrictions d'usage avec les appels spécifiques GSS.
- Adaptation et incorporation de la discussion des liens C sur la comparaison du nom avec les objets Nom exporté.
- Ajout d'une recommandation pour les concepteurs de mécanisme pour la prise en charge d'un type de nom de service fondé sur l'hôte, déférant toute déclaration d'exigence aux spécifications individuelles de mécanisme. Ajout de la discussion de l'élément de nom de service du service fondé sur l'hôte et proposition d'une approche pour la politique

d’enregistrement de l’IANA.
- Précision sur l’ordre des octets au sein de l’objet Nom exporté. Mention que GSS_S_BAD_MECH est à retourner si, dans le cours d’une tentative d’importation d’un objet Nom exporté, le type de mécanisme inclus dans l’objet de nom n’est pas reconnu ou pris en charge.
- Mention que les mécanismes peuvent facultativement accepter GSS_C_NO_NAME comme entrée de nom cible à GSS_Init_sec_context(), avec le commentaire qu’une telle prise en charge est peu probable au sein des mécanismes antérieurs à GSS-v2, mise à jour 1.

Adresse de l’auteur

John Linn
RSA Laboratories
20 Crosby Drive
Bedford, MA 01730 USA
téléphone : +1 781.687.7817
mél : jlinn@rsasecurity.com

Déclaration complète de droits de reproduction

Remerciement

Le financement de la fonction d’édition des RFC est actuellement fourni par l’Internet Society.