ExtraHop 9.9
Référence de l'API Trigger

Pour plus de documentation, voir https://docs.extrahop.com.

Publié: 2025-01-05

ExtraHop Networks
Seattle, WA 98101
877-333-9872 (US)
+44 (0)203 7016850 (EMEA)
+65-31585513 (APAC)
www.extrahop.com

Table des matières

Vue d'ensemble

Les déclencheurs d'inspection des applications sont composés d'un code défini par l'utilisateur qui s' exécute automatiquement en fonction des événements du système via l'API de déclenchement ExtraHop. En écrivant des déclencheurs, vous pouvez collecter des données métriques personnalisées sur les activités de votre réseau. En outre, les déclencheurs peuvent effectuer des opérations sur protocole des messages (tels qu'un HTTP request) avant que le paquet ne soit supprimé.

Le système ExtraHop surveille, extrait et enregistre un ensemble de base de couches 7 (L7) des métriques relatives aux appareils du réseau, telles que le nombre de réponses, le nombre d'erreurs et les temps de traitement. Une fois ces métriques enregistrées pour un protocole L7 donné, les paquets sont supprimés, libérant ainsi des ressources pour la poursuite du traitement.

Les déclencheurs vous permettent de :

- Générez et stockez des métriques personnalisées dans la banque de données interne du système ExtraHop. Par exemple, bien que le système ExtraHop ne collecte pas d'informations sur l'agent utilisateur qui a généré une requête HTTP, vous pouvez générer et collecter ce niveau de détail en écrivant un déclencheur et en validant les données dans la banque de données. Vous pouvez également afficher les données personnalisées stockées dans la banque de données en créant des pages de mesures personnalisées et en affichant ces mesures via l' explorateur de mesures et tableaux de bord.
- Générez et envoyez des enregistrements pour un stockage à long terme et une extraction vers un espace de stockage des enregistrements.
- Créez une application définie par l'utilisateur qui collecte des métriques sur plusieurs types de trafic réseau afin de recueillir des informations ayant un impact sur tous les niveaux. Par exemple, pour obtenir une vue unifiée de l' ensemble du trafic réseau associé à un site Web, des transactions Web aux DNS demandes et réponses aux transactions de base de données : vous pouvez créer une application contenant toutes ces mesures relatives au site Web.
- Générez des métriques personnalisées et envoyez les informations aux utilisateurs de Syslog tels que Splunk, ou à des bases de données tierces telles que MongoDB ou Kafka.
- Lancez une capture de paquets pour enregistrer des flux individuels en fonction de critères définis par l'utilisateur. Vous pouvez télécharger les flux capturés et les traiter à l'aide d'outils tiers. Votre système ExtraHop doit disposer d'une licence pour la capture de paquets pour accéder à cette fonctionnalité.

Le but de ce guide est de fournir des informations de référence lors de l'écriture des blocs de code JavaScript qui s'exécutent lorsque les conditions du déclencheur sont remplies. Le Ressources de l'API de déclenchement Cette section contient une liste de rubriques qui fournissent une vue d' ensemble complète des concepts et procédures relatifs aux déclencheurs.

Ressources de l'API de déclenchement

Cette section contient une liste de rubriques qui vous aideront à vous familiariser avec les concepts de déclencheur, la création d'un déclencheur et les meilleures pratiques.

- éléments déclencheurs
- Créez un déclencheur
  - Configurer les paramètres du déclencheur
  - Écrire un script de déclencheur
- Surveillez les performances du déclencheur
- Guide des meilleures pratiques relatives aux déclencheurs
- FAQ sur les déclencheurs
- Procédure pas à pas : Créez un déclencheur pour collecter des métriques personnalisées pour les erreurs HTTP 404
- Procédure pas à pas : Initiez des captures de paquets de précision pour analyser les conditions de fenêtre zéro
- Procédure pas à pas : Créez un déclencheur pour surveiller les réponses aux requêtes NTP monlist

Types de données pour les métriques personnalisées

L'API ExtraHop Trigger vous permet de créer des métriques personnalisées qui collectent des données sur votre environnement, au-delà de ce qui est fourni par les métriques du protocole intégré.

Vous pouvez créer des métriques personnalisées à partir des types de données suivants :

compter

Le nombre d'événements métriques survenus au cours d'une période donnée. Par exemple, pour enregistrer des informations sur le nombre de requêtes HTTP au fil du temps, sélectionnez une métrique de dénombrement de niveau supérieur. Vous pouvez également sélectionner une métrique de nombre de détails pour enregistrer des informations sur le nombre de fois clients a accédé à un serveur, avec la clé IPAddress et un entier représentant le nombre d'accès sous forme de valeur.

instantané

Type spécial de métrique de comptage qui, lorsqu'elle est interrogée au fil du temps, renvoie la valeur la plus récente (telle que les connexions TCP établies).

distincte

Nombre estimé d'éléments uniques observés au fil du temps, tels que le nombre de ports uniques ayant reçu des paquets SYN, un nombre élevé pouvant indiquer une analyse des ports.

jeu de données

Un résumé statistique des informations temporelles, tel qu'un résumé à 5 chiffres : min, 25e percentile, médiane, 75e percentile, max. Par exemple, pour enregistrer des informations sur le temps de traitement HTTP au fil du temps, sélectionnez un niveau supérieur jeu de données métrique.

ensemble d'échantillons

Résumé statistique des informations temporelles, telles que la moyenne et l'écart type. Par exemple, pour enregistrer des informations sur le temps nécessaire au serveur pour traiter chaque URI, sélectionnez un ensemble d'échantillons détaillé avec la clé de chaîne d'URI et un entier représentant le temps de traitement sous forme de valeur.

max

Type spécial de métrique de comptage qui préserve le maximum. Par exemple, pour enregistrer les instructions HTTP les plus lentes au fil du temps sans vous fier à une table de session, sélectionnez une métrique de niveau supérieur et une métrique maximale de détail.

Les métriques personnalisées sont prises en charge pour les types de sources suivants :

- Application
- Device
- Network
- FlowInterface
- FlowNetwork

Pour plus d'informations sur les différences entre les mesures de haut niveau et les mesures détaillées, consultez le FAQ sur les métriques.

Fonctions globales

Les fonctions globales peuvent être appelées sur n'importe quel événement.

`cache(key:` ***Corde*** `, valueFn: () =>` ***N'importe lequel*** `):` ***N'importe lequel***

Met en cache les paramètres spécifiés dans une table pour permettre une recherche et un renvoi efficaces de grands ensembles de données.

`key:` ***Corde***

Un identifiant qui indique l'emplacement de la valeur mise en cache. Une clé doit être unique au sein d'un déclencheur.

`valueFn: () =>` ***N'importe lequel***

Fonction sans argument qui renvoie une valeur non nulle.

Dans l'exemple suivant, `cache` la méthode est appelée avec de grandes quantités de données codées en dur dans le script déclencheur :

```
let storeLookup = cache("storesByNumber", () => ({
    1 : "Newark",
    2 : "Paul",
    3 : "Newark",
    4 : "St Paul"// 620 lines omitted
}));

var storeCity;
var query = HTTP.parseQuery(HTTP.query);

if (query.storeCode) {
   storeCity = storeLookup[parseInt(query.storeCode)];
}
```

Dans l'exemple suivant, la liste des agents utilisateurs connus dans un déclencheur JBoss est normalisée avant d'être comparée à l'agent utilisateur observé. Le déclencheur convertit la liste en minuscules et supprime les espaces superflus, puis met les entrées en cache.

```
function jbossUserAgents() {
    return [
        // Add your own user agents here, followed by a comma
        "Gecko-like (Edge 14.0; Windows 10; Silverlight or similar)",
        "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5)
 AppleWebKit/537.36
         (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36",
        "Mozilla/5.0 (Android)"
    ].map(ua => ua.trim().toLowerCase());
}

var badUserAgents = cache("badUserAgents", jbossUserAgents);
```

`commitDetection(type:` ***Corde*** `, options:` ***Objet*** `)`

Génère une détection sur le système ExtraHop.

`type:` ***Corde***

Un type défini par l'utilisateur pour la définition, tel que `brute_force_attack`. Tu peux syntoniser les détections pour masquer plusieurs détections du même type. La chaîne ne peut contenir que des lettres, des chiffres et des traits de soulignement.

`options:` ***Objet***

Un objet qui spécifie les propriétés suivantes pour la détection :

`title:` ***Corde***

Titre défini par l'utilisateur qui identifie la détection.

`description:` ***Corde***

Description de la détection.

`riskScore:` ***Numéro*** | ***nul***

Nombre facultatif compris entre 1 et 99 qui représente l'indice de risque de la détection.

`participants:` ***Tableau d'objets***

Un ensemble facultatif d'objets participants associés à la détection. Un objet participant doit contenir les propriétés suivantes :

`object:` ***Objet***

L'objet de l'appareil, de l'application ou de l'adresse IP associé au participant.

`role:` ***Corde***

Le rôle du participant à la détection. Les valeurs suivantes sont valides :

- `offender`
- `victim`

`identityKey:` ***Corde*** | ***nul***

Un identifiant unique qui permet des détections continues. Si plusieurs détections avec la même clé d'identité et le même type de détection sont générées au cours de la période spécifiée par le `identityTtl` propriété, les détections sont consolidées en une seule détection continue.

Note: Si le système ExtraHop génère un grand nombre de détections avec des clés d'identité uniques, il se peut que le système ne parvienne pas à consolider certaines détections en cours. Toutefois, le système ne générera pas plus de 250 détections individuelles pour un déclencheur par jour.

`identityTtl:` ***Corde***

Durée après la génération d'une détection pendant laquelle les détections dupliquées sont consolidées dans une détection continue.

Une fois qu'une détection est générée, si une autre détection avec la même clé d'identité et le même type de détection est générée dans le délai spécifié, les deux détections sont consolidées en une seule détection continue. Chaque fois qu'une détection est consolidée en une détection continue, la période est réinitialisée et la détection ne s'arrête pas avant l'expiration de cette période. Par exemple, si `identityTtl` est réglé sur `day`, et quatre détections en double sont générées chacune à 12 heures d'intervalle, la détection continue s'étalant sur trois jours. Les périodes de validité suivantes sont les suivantes :

- `hour`
- `day`
- `week`

La période par défaut est `hour`.

`commitRecord(id:` ***Corde*** `, record:` ***Objet*** `):` ***vide***

Envoie un objet d'enregistrement personnalisé à l'espace de stockage des enregistrements configuré.

`id:` ***Corde***

L'ID du type d'enregistrement à créer. L'ID ne peut pas commencer par un tilde (~).

`record:` ***Objet***

Objet contenant une liste de paires de propriétés et de valeurs à envoyer à l'espace de stockage des enregistrements configuré en tant qu'enregistrement personnalisé.

Les propriétés suivantes sont automatiquement ajoutées aux enregistrements et ne sont pas représentées sur les objets renvoyés par les accesseurs d'enregistrement intégrés, telles que `HTTP.record`:

- `ex`
- `flowID`
- `client`
- `clientAddr`
- `clientPort`
- `receiver`
- `receiverAddr`
- `receiverPort`
- `sender`
- `senderAddr`
- `senderPort`
- `server`
- `serverAddr`
- `serverPort`
- `timestamp`
- `vlan`

Par exemple, pour accéder à `flowID` propriété dans un enregistrement HTTP, vous devez inclure `HTTP.record.Flow.id` dans votre déclaration.

Important: Pour éviter des données inattendues dans l'enregistrement ou une exception lors de l'appel de la méthode, les noms de propriété répertoriés ci-dessus ne peuvent pas être spécifiés en tant que nom de propriété dans des enregistrements personnalisés.

En outre, le nom d'une propriété figurant dans les enregistrements personnalisés ne peut contenir aucun des caractères suivants :

`.`
: Période

`:`
: Colon

`[`
: Support carré

`]`
: Support carré

Dans l'exemple suivant, les deux paires de propriétés et de valeurs qui ont été ajoutées à `record` les variables sont enregistrées dans un enregistrement personnalisé par `commitRecord` fonction :

```
var record = {
   'field1': myfield1,
   'field2': myfield2
};
commitRecord('record_type_id', record);
```

Pour la plupart des événements, vous pouvez valider un enregistrement intégré contenant les propriétés par défaut. Par exemple, un enregistrement intégré tel que le `HTTP.record` L'objet peut servir de base à un enregistrement personnalisé.

L'exemple de code suivant valide un enregistrement personnalisé qui inclut toutes les métriques intégrées du `HTTP.record` objet et une métrique supplémentaire provenant du `HTTP.headers` propriété :

```
var record = Object.assign(
   {'server': HTTP.headers.server},
   HTTP.record
);
commitRecord('custom-http-record', record);
```

Vous pouvez accéder à un objet d'enregistrement intégré pour les événements suivants :

Classe	Évènements
AAA	AAA_REQUEST AAA_RESPONSE
ActiveMQ	ACTIVEMQ_MESSAGE
AJP	AJP_RESPONSE
CIFS	CIFS_RESPONSE
DB	DB_RESPONSE
DHCP	DHCP_REQUEST DHCP_RESPONSE
DICOM	DICOM_REQUEST DICOM_RESPONSE
DNS	DNS_REQUEST DNS_RESPONSE
FIX	FIX_REQUEST FIX_RESPONSE
Flow	FLOW_RECORD
FTP	FTP_RESPONSE
HL7	HL7_RESPONSE
HTTP	HTTP_RESPONSE
IBMMQ	IBMMQ_REQUEST IBMMQ_RESPONSE
ICA	ICA_OPEN ICA_CLOSE ICA_TICK
ICMP	ICMP_MESSAGE
Kerberos	KERBEROS_REQUEST KERBEROS_RESPONSE
LDAP	LDAP_REQUEST

Classe	Évènements
	LDAP_RESPONSE
Memcache	MEMCACHE_REQUEST MEMCACHE_RESPONSE
Modbus	MODBUS_RESPONSE
MongoDB	MONGODB_REQUEST MONGODB_RESPONSE
MSMQ	MSMQ_MESSAGE
NetFlow	NETFLOW_RECORD
NFS	NFS_RESPONSE
NTLM	NTLM_MESSAGE
POP3	POP3_RESPONSE
RDP	RDP_OPEN RDP_CLOSE RDP_TICK
Redis	REDIS_REQUEST REDIS_RESPONSE
RTCP	RTCP_MESSAGE
RTP	RTP_TICK
SCCP	SCCP_MESSAGE
SFlow	SFLOW_RECORD
SIP	SIP_REQUEST SIP_RESPONSE
SMPP	SMPP_RESPONSE
SMTP	SMTP_RESPONSE
SSH	SSH_OPEN SSH_CLOSE SSH_TICK
SSL	SSL_ALERT SSL_OPEN SSL_CLOSE SSL_HEARTBEAT SSL_RENEGOTIATE
Telnet	TELNET_MESSAGE

`debug(message:` ***Corde*** `):` ***vide***

Ecrit au journal de débogage si le débogage est activé. La taille maximale des messages est de 2 048 octets. Les messages de plus de 2 048 octets sont tronqués.

`getTimestamp():` ***Numéro***

Renvoie l'horodateur du paquet qui a provoqué l'exécution de l'événement déclencheur, exprimé en millisecondes, les microsecondes étant le segment fractionnaire après la virgule.

`log(message:` ***Corde*** `):` ***vide***

Écrit dans le journal de débogage, que le débogage soit activé ou non.

Les appels multiples pour déboguer et enregistrer des instructions dans lesquelles le message a la même valeur seront affichés toutes les 30 secondes.

La limite des entrées du journal de débogage est de 2 048 octets. Pour enregistrer des entrées plus volumineuses, voir Remote.Syslog .

`md5(message:` ***Corde*** | ***Tampon*** `):` ***Corde***

Hache la représentation UTF-8 du message spécifié Tampon objet ou chaîne et renvoie la somme MD5 de la chaîne.

`sha1(message:` ***Corde*** | ***Tampon*** `):` ***Corde***

Hache la représentation UTF-8 du message spécifié Tampon objet ou chaîne et renvoie la somme SHA-1 de la chaîne.

`sha256(message:` ***Corde*** | ***Tampon*** `):` ***Corde***

Hache la représentation UTF-8 du message spécifié Tampon objet ou chaîne et renvoie la somme SHA-256 de la chaîne.

`sha512(message:` ***Corde*** | ***Tampon*** `):` ***Corde***

Hache la représentation UTF-8 du message spécifié Tampon objet ou chaîne et renvoie la somme SHA-512 de la chaîne.

`uuid():` ***Corde***

Renvoie un identifiant unique universel (UUID) aléatoire version 4.

Cours à usage général

Les classes de l'API Trigger présentées dans cette section fournissent des fonctionnalités largement applicables à tous les événements.

Classe	Descriptif
Application	Vous permet de créer de nouvelles applications et d'ajouter des métriques personnalisées au niveau de l'application.
Tampon	Vous permet d'accéder au contenu de la mémoire tampon.
Detection	Vous permet de récupérer des informations sur les détections sur le système ExtraHop.
Device	Vous permet de récupérer les attributs de l'équipement et d'ajouter des mesures personnalisées au niveau de l'équipement.
Discover	Vous permet d'accéder aux appareils et applications récemment découverts.
Flow	Le flux fait référence à une conversation entre deux points de terminaison via un protocole tel que TCP, UDP ou ICMP. La classe Flow permet d'accéder à des éléments de ces conversations, tels que les adresses IP des points de terminaison et l'âge du flux. La classe Flow contient également un magasin de flux conçu pour transmettre des objets d'une demande à une réponse sur le même flux.
FlowInterface	Vous permet de récupérer les attributs de l'interface de flux et d'ajouter des métriques personnalisées au niveau de l' interface.
FlowNetwork	Vous permet de récupérer les attributs du réseau de flux et d'ajouter des métriques personnalisées au niveau du réseau de flux.
GeoIP	Vous permet de récupérer l'emplacement approximatif d'une adresse IP spécifique au niveau du pays ou de la ville.
IPAddress	Permet de récupérer les attributs d'adresse IP.
Network	Vous permet d'ajouter des métriques personnalisées au niveau global.
Session	Vous permet d'accéder au tableau des sessions, qui permet la coordination entre plusieurs déclencheurs s'exécutant indépendamment.
System	Vous permet d'accéder aux propriétés qui identifient le système ExtraHop sur lequel un déclencheur est exécuté.
ThreatIntel	Vous permet de voir si une adresse IP, un nom d'hôte ou un URI est suspect.

Classe	Descriptif
Trigger	Vous permet d'accéder aux informations relatives à un déclencheur en cours d'exécution.
VLAN	Vous permet d'accéder aux informations relatives à un VLAN sur le réseau.

`Application`

Le `Application` La classe vous permet de collecter des métriques sur plusieurs types de trafic réseau afin de capturer des informations ayant un impact à plusieurs niveaux. Par exemple, si vous souhaitez obtenir une vue unifiée de l'ensemble du trafic réseau associé à un site Web (des transactions Web aux requêtes DNS, en passant par les réponses aux transactions de base de données), vous pouvez créer un déclencheur pour créer une application personnalisée contenant toutes ces mesures connexes. Le `Application` La classe vous permet également de créer des métriques personnalisées et de valider les données métriques dans des applications. Les applications ne peuvent être créées et définies que par le biais de déclencheurs.

Méthodes d'instance

Les méthodes de cette section ne peuvent pas être appelées directement sur `Application` classe. Vous ne pouvez appeler ces méthodes que sur des instances de classe Application spécifiques. Par exemple, la déclaration suivante est valide :

```
Application("sampleApp").metricAddCount("responses", 1);
```

Toutefois, la déclaration suivante n'est pas valide :

```
Application.metricAddCount("responses", 1);
```

`commit(id:` ***Corde*** `):` ***vide***

Crée une application, valide les métriques intégrées associées à l'événement dans l' application et ajoute l'application à tous les enregistrements intégrés ou personnalisés enregistrés pendant l'événement.

L'ID de l'application doit être une chaîne. Pour les métriques intégrées aux applications, les métriques ne sont validées qu'une seule fois, même si `commit()` La méthode est appelée plusieurs fois lors du même événement.

L'instruction suivante crée une application nommée « MyApp » et valide les métriques intégrées dans l' application :

```
Application("myApp").commit();
```

Si vous envisagez de valider des métriques personnalisées dans une application, vous pouvez créer l'application sans appeler le `commit()` méthode. Par exemple, si l'application n'existe pas encore, l'instruction suivante crée l'application et valide la métrique personnalisée dans l' application :

```
Application("myApp").metricAddCount("requests", 1);
```

Vous pouvez appeler le `Application.commit` méthode uniquement sur les événements suivants :

Types de métriques	Événement
AAA	AAA_REQUEST -et- AAA_RESPONSE
AJP	AJP_RESPONSE

Types de métriques	Événement
CIFS	CIFS_RESPONSE
DB	DB_RESPONSE
DHCP	DHCP_REQUEST -et- DHCP_RESPONSE
DNS	DNS_REQUEST -et- DNS_RESPONSE
FIX	FIX_REQUEST -et- FIX_RESPONSE
FTP	FTP_RESPONSE
HTTP	HTTP_RESPONSE
IBMMQ	IBMMQ_REQUEST -et- IBMMQ_RESPONSE
ICA	ICA_TICK -et- ICA_CLOSE
Kerberos	KERBEROS_REQUEST -et- KERBEROS_RESPONSE
LDAP	LDAP_REQUEST -et- LDAP_RESPONSE
Memcache	MEMCACHE_REQUEST -et- MEMCACHE_RESPONSE
Modbus	MODBUS_RESPONSE
MongoDB	MONGODB_REQUEST -et- MONGODB_RESPONSE
NAS	CIFS_RESPONSE -et/ou- NFS_RESPONSE
NetFlow	NETFLOW_RECORD Notez que la validation n'aura pas lieu si des identifiants d'entreprise sont présents dans l'enregistrement NetFlow.
NFS	NFS_RESPONSE
RDP	RDP_TICK
Redis	REDIS_REQUEST -et- REDIS_RESPONSE
RPC	RPC_REQUEST -et- RPC_RESPONSE
RTP	RTP_TICK
RTCP	RTCP_MESSAGE
SCCP	SCCP_MESSAGE
SIP	SIP_REQUEST -et- SIP_RESPONSE
SFlow	SFLOW_RECORD
SMTP	SMTP_RESPONSE
SSH	SSH_CLOSE -et- SSH_TICK
SSL	SSL_RECORD -et- SSL_CLOSE
WebSocket	WEBSOCKET_OPEN, WEBSOCKET_CLOSE, et WEBSOCKET_MESSAGE

`metricAddCount(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de comptage. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Le nom de la métrique de comptage de niveau supérieur.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailCount(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de comptage par lequel vous pouvez approfondir. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDataset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de l'ensemble de données. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Nom de la métrique de l'ensemble de données de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailDataset(metric_name:` ***Corde*** `, key:` ***Corde*** | ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de l'ensemble de données par lequel vous pouvez approfondir. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDistinct(metric_name:` ***Corde*** `, item:` ***Numéro*** | ***Corde*** | ***Adresse IP*** `:void`

Crée une personnalisation niveau supérieur métrique de comptage distincte. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Nom de la métrique de comptage distincte de niveau supérieur.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

`metricAddDetailDistinct(metric_name:` ***Corde*** `, key:` ***Corde*** | ***Adresse IP*** `, item:` ***Numéro*** | ***Corde*** | ***Adresse IP*** `:void`

Crée une personnalisation détail métrique de comptage distincte par lequel vous pouvez approfondir. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Nom de la métrique de comptage distincte détaillée.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

`metricAddMax(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique maximale. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Le nom de la métrique maximale de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailMax(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique maximale par lequel vous pouvez approfondir. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Nom de la métrique maximale de détail.

`key:` ***Corde | Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSampleset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur Sampleset métrique. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Le nom de la métrique de l'ensemble d'échantillons de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSampleset(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail Sampleset métrique par lequel vous pouvez approfondir. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde | Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

highPrecision: ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSnap(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de capture d'écran. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Nom de la métrique de capture instantanée de niveau supérieur.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSnap(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de capture d'écran par lequel vous pouvez approfondir. Valide les données métriques dans l'application spécifiée.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`toString():` ***Corde***

Renvoie l'objet Application sous forme de chaîne au format suivant :

```
[object Application <application_id>]
```

Propriétés de l'instance

`id:` ***Corde***

L'identifiant unique de l'application, tel qu'indiqué dans le système ExtraHop sur la page de cette application.

Exemples de déclencheurs

- Exemple : création d'un conteneur d'applications

Tampon

Le `Buffer` La classe donne accès à des données binaires.

Un tampon est un objet présentant les caractéristiques d'un tableau. Chaque élément du tableau est un nombre compris entre 0 et 255, représentant un octet. Chaque objet tampon possède une propriété de longueur (le nombre d'éléments dans un tableau) et un opérateur entre crochets.

La charge utile chiffrée n'est pas déchiffrée pour l'analyse de la charge utile TCP et UDP.

UDP_PAYLOAD nécessite une chaîne correspondante mais TCP_PAYLOAD ne le fait pas. Si vous ne spécifiez pas de chaîne correspondante pour TCP_PAYLOAD, le déclencheur s'exécute une fois après les N premiers octets de charge utile.

Méthodes

`Buffer(string:` ***Corde*** `| format:` ***Corde*** `)`

Constructeur de la classe Buffer qui décode une chaîne codée en un objet Buffer. Les paramètres suivants sont obligatoires :

`string:` ***Corde***

La chaîne codée.

`format:` ***Corde***

Format dans lequel l'argument de chaîne est encodé. Les formats de codage suivants sont valides :

- `base64`
- `base64url`

Méthodes d'instance

`decode(type:` ***Corde*** `):` ***Corde***

Interprète le contenu de la mémoire tampon et renvoie une chaîne avec l'une des options suivantes :

- `utf-8`
- `utf-16`
- `ucs2`
- `hex`

`equals(buffer:` ***Tampon*** `):` ***Booléen***

Effectue un test d'égalité entre les objets Buffer, où `buffer` est l' objet à comparer.

`slice(start:` ***Numéro*** `, end:` ***Numéro*** `):` ***Tampon***

Renvoie les octets spécifiés dans un tampon sous la forme d'un nouveau tampon. Les octets sont sélectionnés à partir de l'argument de début donné et se terminant à l'argument de fin (sans inclure).

`start:` ***Numéro***

Entier qui indique où commencer la sélection. Spécifiez les nombres négatifs à sélectionner à la fin d'une zone tampon. Il s'agit d'une base zéro.

`end:` ***Numéro***

Nombre entier facultatif qui indique où terminer la sélection. En cas d'omission, tous les éléments situés entre la position de départ et la fin de la zone tampon seront sélectionnés. Spécifiez les nombres négatifs à sélectionner à la fin d'une zone tampon. Il s'agit d'une base zéro.

`toString(format:` ***Corde*** `):` ***Corde***

Convertit le buffer en chaîne. Le paramètre suivant est facultatif :

`format:` ***Corde***

Le format avec lequel encoder la chaîne. Si aucun encodage n'est spécifié, la chaîne n'est pas encodée. Les formats de codage suivants sont valides :

- `base64`
- `base64url`
- `hex`

`unpack(format:` ***Corde*** `, offset:` ***Numéro*** `):` ***Array***

Traite les données binaires ou à largeur fixe à partir de n'importe quel objet tampon, tel que celui renvoyé par `HTTP.payload`, `Flow.client.payload`, ou `Flow.sender.payload`, selon la chaîne de format donnée et, éventuellement, selon le décalage spécifié.

Renvoie un tableau JavaScript contenant un ou plusieurs champs décompressés et contenant la position absolue en octets de charge utile +1 du dernier octet de l'objet décompressé. La valeur en octets peut être spécifiée comme décalage lors d' appels ultérieurs pour décompresser un tampon.

Note:
- Le `buffer.unpack` La méthode interprète les octets dans l'ordre big-endian par défaut. Pour interpréter les octets dans l'ordre little-endian, préfixez la chaîne de format avec un signe inférieur à (<).
- Le format ne doit pas nécessairement consommer la totalité de la mémoire tampon.
- Les octets nuls ne sont pas inclus dans les chaînes décompressées. Par exemple : `buf.unpack('4s')[0] - > 'example'`.
- Le caractère de format z représente des chaînes de longueur variable terminées par des valeurs nulles. Si le dernier champ est z, la chaîne est produite, que le caractère nul soit présent ou non.
- Une exception est déclenchée lorsque tous les champs ne peuvent pas être décompressés car la mémoire tampon ne contient pas suffisamment de données.

Le tableau ci-dessous répertorie les formats de chaîne de mémoire tampon pris en charge :

Formater	Type C	Type de JavaScript	Taille standard
x	pad type	aucune valeur	
A	struct in6_addr	IPAddress	16
a	struct in_addr	IPAddress	4
b	signed char	string of length 1	1
B	unsigned char	number	1
?	_Bool	boolean	1
H	court métrage non signé	number	2
h	short	number	2
i	int	number	4
I	unsigned int	number	4
l	long	number	4
L	unsigned long	number	4
q	long long	number	8

Formater	Type C	Type de JavaScript	Taille standard
Q	unsigned long long	number	8
f	number	number	4
d	double	number	4
s	char[]	string	
z	char[]	string	

Propriétés de l'instance

`length:` ***Numéro***

Le nombre d'octets dans la mémoire tampon.

Exemples de déclencheurs

- Exemple : analyse NTP avec analyse de charge utile universelle
- Exemple : analyse du syslog sur TCP avec une analyse de charge utile universelle

Detection

Le `Detection` class vous permet de récupérer des informations sur les détections sur le système ExtraHop.

Note: Les détections par apprentissage automatique nécessitent connexion aux services cloud ExtraHop.

Évènements

`DETECTION_UPDATE`

S'exécute lorsqu'une détection est créée ou mise à jour sur le système ExtraHop.

Conseil: Au lieu d'écrire un déclencheur pour exporter les données de détection, nous vous recommandons créer une règle de notification de détection. Vous pouvez configurer ces règles pour envoyer des charges utiles JSON avec un webhook et éviter la complexité liée à l'écriture d'un déclencheur.

Important: Cet événement s'exécute pour toutes les détections, quel que soit l'accès au module accordé à l'utilisateur qui crée le déclencheur. Par exemple, les déclencheurs créés par les utilisateurs ayant accès au module NPM s'exécutent sur `DETECTION_UPDATE` événements pour les détections de sécurité et de performance.

Note: Cet événement ne s'exécute pas lorsque l'état d'un ticket de détection est mis à jour. Par exemple, la modification d'un responsable de la détection ne provoquera pas l'exécution de l'événement DETECTION_UPDATE. Cet événement ne s'exécute pas non plus pour les détections masquées.

Note: Vous ne pouvez pas attribuer des déclencheurs qui s'exécutent uniquement lors de cet événement à des appareils ou à des groupes d'équipements spécifiques. Les déclencheurs qui s'exécutent lors de cet événement seront exécutés chaque fois que cet événement se produira.

Propriétés

`applianceId:` ***Numéro***

Si vous faites appel à console, renvoie l'ID de la sonde connectée sur laquelle la détection a eu lieu. En cas d'appel sur une sonde, renvoie `0`.

`assignee:` ***Corde***

L'assigné du ticket associé à la détection.

`categories:` ***Tableau de cordes***

Liste des catégories auxquelles appartient la détection.

Les valeurs suivantes sont valides :

Valeur	Catégorie
`sec`	Sûreté
`sec.action`	Actions par rapport à l'objectif
`sec.botnet`	botnet
`sec.caution`	Mise en garde
`sec.command`	Commandement et contrôle
`sec.cryptomining`	Cryptominage
`sec.dos`	Déni de service
`sec.exfil`	Exfiltration
`sec.exploit`	Exploitation
`sec.hardening`	Durcissement
`sec.lateral`	Mouvement latéral
`sec.ransomware`	Un ransomware
`sec.recon`	Reconnaissance
`perf`	Rendement
`perf.auth`	Autorisation et contrôle d'accès
`perf.db`	Base de données
`perf.network`	Infrastructure réseau
`perf.service`	Dégradation du service
`perf.storage`	Rangement
`perf.virtual`	Virtualisation des ordinateurs de bureau et des applications
`perf.web`	Application Web

`description:` ***Corde***

Description de la détection.

Conseil Il est souvent plus facile d'extraire des informations relatives à une détection à partir du `Detection.properties` propriété plutôt que d'analyser la `Detection.description` texte. Pour plus d'informations, consultez le `Detection.properties` description.

Le tableau suivant répertorie les formats Markdown courants que vous pouvez inclure dans la description :

Formater	Descriptif	Exemple
Rubriques	Placez un signe numérique (#) et un espace devant votre texte pour mettre en forme les titres. Le niveau du titre est déterminé par le nombre de signes numériques.	`#### Example H4 heading`
Listes non ordonnées	Placez un astérisque (*) avant votre texte. Si possible, placez chaque élément de la liste sur une ligne distincte.	`* First example` `* Second example`
Listes ordonnées	Placez le chiffre 1 et le point (1.) avant votre texte pour chaque élément de ligne ; Markdown incrémentera automatiquement le numéro de liste. Si possible, placez chaque élément de la liste sur une ligne distincte.	`1. First example` `1. Second example`
AUDACIEUX	Placez un double astérisque avant et après votre texte.	`**bold text**`
Italiques	Placez un trait de soulignement avant et après votre texte.	`_italicized text_`
Hyperliens	Placez le texte du lien entre crochets avant l'URL entre parenthèses. Ou saisissez votre URL. Les liens vers des sites Web externes s'ouvrent dans un nouvel onglet du navigateur. Les liens du système ExtraHop, tels que les tableaux de bord, s'ouvrent dans l'onglet actuel du navigateur.	`[Visit our home page](https://www.extrahop.com)` `https://www.extrahop.com`
Citations en blocs	Placez un crochet à angle droit et un espace devant votre texte.	`On the ExtraHop website:` `> Access the live demo and review case studies.`
Emojis	Copiez et collez une image emoji dans la zone de texte. Consultez les Graphique Emoji Unicode site web pour les images.	

Formater	Descriptif	Exemple
	La syntaxe Markdown ne prend pas en charge les shortcodes emoji.	

`endTime:` ***Numéro***

Heure à laquelle la détection s'est terminée, exprimée en millisecondes depuis l'époque.

`id:` ***Numéro***

L'identifiant unique pour la détection.

`isCustom:` ***Booléen***

La valeur est `true` s'il s'agit d'une détection personnalisée générée par un déclencheur.

`isEventCreate:` ***Booléen***

Si la valeur est vraie, `DETECTION_UPDATE` événement exécuté lors de la création de la détection. Si la valeur est fausse, `DETECTION_UPDATE` événement exécuté lors de la mise à jour de la détection.

`mitreCategories:` ***Tableau d'objets***

Ensemble d'objets contenant les techniques et tactiques MITRE associées à la détection. Chaque objet contient les propriétés suivantes :

`id`

L'identifiant de la technique ou de la tactique MITRE.

`name`

Nom de la technique ou de la tactique MITRE.

`url`

Adresse Web de la technique ou de la tactique sur le site Web du MITRE.

`participants:` ***Tableau d'objets***

Un ensemble d'objets participants associés à la détection. Un objet participant contient les propriétés suivantes :

`object:` ***Objet***

L'appareil, l'application ou l'objet d'adresse IP associé au participant.

`id:` ***Numéro***

L'identifiant du participant.

`role:` ***Corde***

Le rôle du participant dans la détection. Les valeurs suivantes sont valides :

- `offender`
- `victim`

`properties:` ***Objet***

Objet contenant les propriétés de la détection. Seuls les types de détection intégrés incluent des propriétés de détection. Le type de détection détermine les propriétés disponibles.

Les noms de champs de l'objet sont les noms des propriétés de détection. Par exemple, le type de détection Anonymous FTP Auth Enabled inclut `client_port` propriété, à laquelle vous pouvez accéder à l'aide du code suivant :

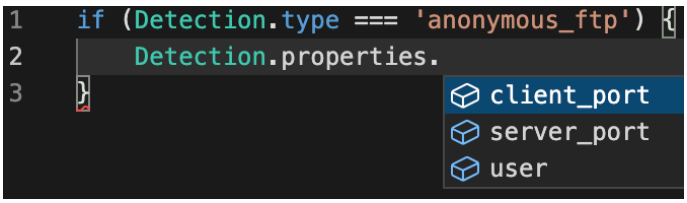
```
Detection.properties.client_port
```

Pour afficher les noms des propriétés de détection, affichez les types de détection à l'aide du `GET /detections/formats` opération dans l'API REST ExtraHop.

Conseil Dans l'éditeur de déclencheurs, vous pouvez afficher les propriétés de détection valides à l'aide de la fonctionnalité de saisie semi-automatique si vous incluez une

logique qui détermine le type de détection. Par exemple, si le déclencheur contient le code suivant et que vous saisissez un point après « propriétés », l'éditeur de déclencheurs affiche les propriétés valides pour la détection Anonymous FTP Auth Enabled :

```
if (Detection.type === 'anonymous_ftp') {
    Detection.properties
}
```



`resolution:` ***Corde***

Résolution du ticket associé à la détection. Les valeurs valides sont `action_taken` et `no_action_taken`.

`riskScore:` ***nombre*** | ***nul***

L'indice de risque de la détection.

`startTime:` ***Numéro***

Heure à laquelle la détection a commencé, exprimée en millisecondes depuis l'époque.

`status:` ***Corde***

État du ticket associé à la détection. Les valeurs valides sont `acknowledged`, `new`, `in_progress`, et `closed`.

`ticketId:` ***Corde***

L'ID du ticket associé à la détection.

`title:` ***Corde***

Titre de la détection.

`type:` ***Corde***

Type de détection. Pour les détections personnalisées, « custom » est ajouté au début de la chaîne définie par l' utilisateur. Par exemple, si vous spécifiez `brute_force_attack` dans le `commitDetection` fonction, le type de détection est `custom.brute_force_attack`.

`updateTime:` ***Numéro***

Dernière mise à jour de la détection, exprimée en millisecondes depuis l' époque.

`Device`

Le `Device` La classe vous permet de récupérer les attributs de l'équipement et d'ajouter des mesures personnalisées au niveau de l'équipement.

Méthodes

`Device(id:` ***Corde*** `)`

Constructeur de l'objet Device qui accepte un paramètre, qui est un identifiant de chaîne unique de 16 caractères.

S'il est fourni avec un identifiant provenant d'un objet Device existant, le constructeur crée une copie de cet objet avec toutes les propriétés de l'objet, comme illustré dans l' exemple suivant :

```
myDevice = new Device(Flow.server.device.id);
debug("myDevice MAC: " + myDevice.hwaddr);
```

Métriques associées à un objet Device par le biais d'un `metricAdd*` les fonctions sont conservées dans la banque de données

`lookupByIP(addr:` ***Adresse IP*** `|` ***Corde*** `, vlan:` ***Numéro*** `):` ***Appareil***

Renvoie l'équipement L3 qui correspond à l'adresse IP et à l'ID de VLAN spécifiés. Retours `null` si aucune correspondance n'est trouvée.

`addr:` ***Adresse IP*** `|` ***Corde***

L'adresse IP de l'équipement. L'adresse IP peut être spécifiée en tant que IPAddress objet ou sous forme de chaîne.

`vlan:` ***nombre***

L'ID VLAN de l'équipement. Renvoie une valeur par défaut de `0` si aucun ID de VLAN n'est fourni ou si la valeur du `devices_across_vlans` les paramètres sont définis sur `true` dans le fichier de configuration en cours d'exécution.

`lookupByMAC(addr:` ***Corde*** `, vlan:` ***Numéro*** `):` ***Appareil***

Renvoie l'équipement L2 qui correspond à l'adresse MAC et à l'ID VLAN spécifiés. Retours `null` si aucune correspondance n'est trouvée.

`addr:` ***Corde***

L'adresse MAC de l'équipement.

`vlan:` ***Numéro***

L'ID VLAN de l'équipement. Renvoie une valeur par défaut de `0` si aucun ID de VLAN n'est fourni ou si la valeur du `devices_across_vlans` les paramètres sont définis sur `true` dans le fichier de configuration en cours d'exécution.

`toString():` ***Corde***

Renvoie l'objet Device sous forme de chaîne au format suivant :

```
[object Device <discovery_id>]
```

Méthodes d'instance

Les méthodes décrites dans cette section ne sont présentes que sur les instances de la classe Device. La plupart des méthodes vous permettent de créer des mesures personnalisées au niveau de l'appareil, comme illustré dans l'exemple suivant :

```
Flow.server.device.metricAddCount("slow_rsp", 1);
```

Note: Un équipement peut parfois agir en tant que client et parfois en tant que serveur sur un flux.

- Appelez une méthode en tant que `Device.metricAdd*` pour collecter des données pour les deux rôles de l'équipement.
- Appelez une méthode en tant que `Flow.client.device.metricAdd*` pour collecter des données uniquement pour le rôle client, que le déclencheur soit attribué au client ou au serveur.
- Appelez une méthode en tant que `Flow.server.device.metricAdd*` pour collecter des données uniquement pour le rôle de serveur, que le déclencheur soit attribué au client ou au serveur.

`equals(device:` ***Appareil*** `):` ***Booléen***

Effectue un test d'égalité entre les objets Device, où `device` est l' objet à comparer.

`metricAddCount(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de comptage. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Le nom de la métrique de comptage de niveau supérieur.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailCount(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de comptage par lequel vous pouvez approfondir. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde | Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDataset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de l'ensemble de données. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Nom de la métrique du jeu de données de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailDataset(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de l'ensemble de données par lequel vous pouvez approfondir. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDistinct(metric_name:` ***Corde*** `, item:` ***Numéro*** | ***Corde*** | ***Adresse IP*** `:void`

Crée une personnalisation niveau supérieur métrique de comptage distincte. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Le nom de la métrique de comptage distincte de niveau supérieur.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

`metricAddDetailDistinct(metric_name:` ***Corde*** `, key:` ***Corde*** | ***Adresse IP*** `, item:` ***Numéro*** | ***Corde*** | ***Adresse IP*** `:void`

Crée une personnalisation détail métrique de comptage distincte par lequel vous pouvez approfondir. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Nom de la métrique de comptage distincte détaillée.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

`metricAddMax(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique maximale. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Le nom de la métrique maximale de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailMax(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique maximale par lequel vous pouvez approfondir. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Nom de la métrique maximale de détail.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSampleset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur Sampleset métrique. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Le nom de la métrique de l'ensemble d'échantillons de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSampleset(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail Sampleset métrique par lequel vous pouvez approfondir. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSnap(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de capture d'écran. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Nom de la métrique de capture instantanée de niveau supérieur.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSnap(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de capture d'écran par lequel vous pouvez approfondir. Valide les données métriques vers l'équipement spécifié.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

Propriétés de l'instance

Les propriétés suivantes vous permettent de récupérer les attributs de l'équipement et ne sont présentes que sur les instances de la classe Device.

`cdpName:` ***Corde***

Le nom CDP associé à l'équipement, s'il est présent.

`dhcpName:` ***Corde***

Le DHCP nom associé à l'équipement, le cas échéant.

`discoverTime:` ***Numéro***

La dernière fois que le processus de capture a découvert l'équipement (et non l' heure de découverte initiale), exprimée en millisecondes depuis l'époque (1er janvier 1970). Les appareils découverts précédemment peuvent être redécouverts par le processus de capture s'ils deviennent inactifs puis redeviennent actifs, ou si le processus de capture est redémarré.

Pour demander à un déclencheur de s'exécuter uniquement lors de la découverte initiale d'un équipement, consultez le `NEW_DEVICE` événement discuté dans le Discover classe.

`dnsNames:` ***Array***

Tableau de chaînes répertoriant les noms DNS associés à l'équipement, le cas échéant.

`hasTrigger:` ***Booléen***

La valeur est `true` si un déclencheur attribué à l'objet Device est en cours d'exécution.

Si le déclencheur est exécuté sur un événement associé à un Flow objet, le `hasTrigger` la valeur de la propriété est `true` sur au moins l'un des objets Device du flux.

Le `hasTrigger` cette propriété est utile pour distinguer les rôles des équipements. Par exemple, si un déclencheur est attribué à un groupe de serveurs proxy, vous pouvez facilement déterminer si un équipement agit en tant que client ou en tant que serveur, plutôt que de vérifier les adresses IP ou les identifiants des appareils, comme dans l'exemple suivant :

```
//Event: HTTP_REQUEST
if (Flow.server.device.hasTrigger) {
    // Incoming request
} else {
    // Outgoing request
}
```

`hwaddr:` ***Corde***

L'adresse MAC de l'équipement, le cas échéant.

`id:` ***Corde***

L'identifiant unique à 16 caractères de l'équipement, tel qu'indiqué dans le système ExtraHop sur la page de cet appareil.

`ipaddrs:` ***Array***

Une gamme de IPAddress objets représentant les adresses IP connues de l' appareil. Pour L3 appareils, la matrice contient toujours une adresse IP.

`isGateway:` ***Booléen***

La valeur est `true` si l'équipement est une passerelle.

`isL3:` ***Booléen***

La valeur est `true` si l'équipement est un L3 équipement pour enfants.

Important: Si vous n'avez pas activé le système ExtraHop pour découvrir les appareils par adresse IP, la propriété iS3 est toujours définie sur False car le système ne fait pas de distinction entre les appareils enfants L3 et parents L2.

`netbiosName:` ***Corde***

Le nom NetBIOS associé à l'équipement, s'il est présent.

`vlanId:` ***Numéro***

L'ID VLAN de l'équipement.

Exemples de déclencheurs

- Exemple : Surveiller les actions des PME sur les appareils
- Exemple : suivi des réponses HTTP de niveau 500 par ID client et URI
- Exemple : collecte des mesures de réponse sur les requêtes de base de données
- Exemple : envoyer les données de l'équipement découvert à un serveur Syslog distant
- Exemple : accéder aux attributs d'en-tête HTTP
- Exemple : enregistrer les succès et les échecs de Memcache
- Exemple : analyse des clés de cache mémoire
- Exemple : analyse de messages PoS personnalisés avec une analyse de charge utile universelle
- Exemple : ajouter des métriques au magasin du cycle métrique

Discover

Le `Discover` La classe vous permet de récupérer des informations sur les appareils et applications récemment découverts.

Évènements

`NEW_APPLICATION`

S'exécute lorsqu'une application est découverte pour la première fois. Cet événement consomme des ressources de capture.

Note: Vous ne pouvez pas attribuer des déclencheurs qui s'exécutent uniquement lors de cet événement à des appareils ou à des groupes d'équipements spécifiques. Les déclencheurs qui s'exécutent lors de cet événement seront exécutés chaque fois que cet événement se produira.

`NEW_DEVICE`

S'exécute lorsque l'activité est observée pour la première fois sur un équipement. Cet événement consomme des ressources de capture.

Note: Vous ne pouvez pas attribuer des déclencheurs qui s'exécutent uniquement lors de cet événement à des appareils ou à des groupes d'équipements spécifiques. Les déclencheurs qui s'exécutent lors de cet événement seront exécutés chaque fois que cet événement se produira.

Propriétés

`application:` ***Demande***

Une application récemment découverte.

S'applique uniquement à `NEW_APPLICATION` événements.

`device:` ***Appareil***

Un équipement récemment découvert.

S'applique uniquement à `NEW_DEVICE` événements.

Note: Vous ne pouvez pas spécifier cette propriété en tant que participant au `commitDetection` fonction.

Exemples de déclencheurs

- Exemple : envoyer les données de l'équipement découvert à un serveur Syslog distant

ExternalData

Le `ExternalData` class vous permet de récupérer les données envoyées depuis des sources externes à l'API Trigger via l'API REST ExtraHop.

Évènements

`EXTERNAL_DATA`

S'exécute chaque fois que des données sont envoyées au système ExtraHop via le Déclencheurs POST/données externes opération.

Propriétés

`body:` ***Corde***

Les données externes envoyées au déclencheur.

`type:` ***Corde***

Identifiant qui décrit les données envoyées au déclencheur. Le type est défini lorsque les données sont envoyées à l'API REST ExtraHop.

`Flow`

Le flux fait référence à une conversation entre deux terminaux sur un protocole tels que TCP, UDP ou ICMP. Le `Flow` class donne accès à des éléments de ces conversations, tels que les adresses IP des points de terminaison et l'âge du flux. La classe Flow contient également un magasin de flux conçu pour transmettre des objets de la demande à la réponse sur le même flux.

Note: Vous pouvez appliquer la classe Flow sur la plupart L7 événements de protocole, mais il n' est pas pris en charge pour les événements de session ou de banque de données.

Évènements

Si un flux est associé à un ExtraHop surveillé L7 protocole, les événements qui sont en corrélation avec le protocole seront exécutés en plus des événements de flux. Par exemple, un flux associé à HTTP exécutera également le `HTTP_REQUEST` et `HTTP_RESPONSE` événements.

`FLOW_CLASSIFY`

S'exécute chaque fois que le système ExtraHop classe initialement un flux comme étant associé à un protocole spécifique.

Note: Pour les flux TCP, `FLOW_CLASSIFY` l'événement se déroule après le `TCP_OPEN` événement.

Grâce à une combinaison de L7 analyse de la charge utile, observation des poignées de main TCP et heuristique basée sur les numéros de port , le `FLOW_CLASSIFY` l'événement identifie le protocole L7 et les rôles des équipements pour les points de terminaison d'un flux tel que client/serveur ou expéditeur/récepteur.

La nature d'un flux peut changer au cours de sa durée de vie, par exemple en cas de tunneling via HTTP ou de passage du SMTP au SMTP-TLS. Dans ces cas, `FLOW_CLASSIFY` s'exécute à nouveau après le changement de protocole.

Le `FLOW_CLASSIFY` cet événement est utile pour lancer une action sur un flux sur la base de la connaissance la plus précoce des informations de flux, telles que le protocole L7, les adresses IP client/serveur ou les ports expéditeur/récepteur.

Actions communes initiées le `FLOW_CLASSIFY` inclure le démarrage d' une PCAP via `captureStart()` procédé ou association du flux à un contenant d'application via le `addApplication()` méthode.

Des options supplémentaires sont disponibles lorsque vous créez un déclencheur qui s'exécute sur cet événement. Par défaut, `FLOW_CLASSIFY` ne s'exécute pas à l' expiration du flux ; toutefois, vous pouvez configurer un déclencheur pour le faire afin de cumuler des métriques pour les flux qui n'étaient pas classés avant leur expiration. Voir Options de déclencheur avancées pour plus d'informations.

`FLOW_DETACH`

S'exécute lorsque l'analyseur a rencontré une erreur inattendue ou est à court de mémoire et cesse de suivre le flux. De plus, un flux de données de faible qualité avec des paquets manquants peut provoquer le détachement de l'analyseur.

Le `FLOW_DETACH` cet événement est utile pour détecter le contenu malveillant envoyé par clients et serveurs. Voici un exemple de la façon dont un déclencheur peut détecter une erreur DNS réponses sur `FLOW_DETACH` événements :

```
if (event == "FLOW_DETACH" && Flow.l7proto== "DNS") {
    Flow.addApplication("Malformed DNS");
}
```

`FLOW_RECORD`

Permet l'enregistrement des informations relatives à un flux à des intervalles chronométrés. Après `FLOW_CLASSIFY` a couru, le `FLOW_RECORD` cet événement aura lieu tous les *N* secondes et chaque fois qu'un flux se ferme. La valeur par défaut pour *N*, appelé intervalle de publication, est de 30 minutes ; la valeur minimale est de 60 secondes. Vous pouvez définir l'intervalle de publication dans les paramètres d'administration.

`FLOW_TICK`

Vous permet d'enregistrer des informations sur un flux par quantité de données ou par tour. Le `FLOW_TICK` cet événement aura lieu tous les `FLOW_TURN` ou tous les 128 paquets, selon la première éventualité. Également, L2 les données sont réinitialisées à chaque `FLOW_TICK` événement qui vous permet d'additionner des données à chaque coche. Si vous comptez le débit, collectez les données auprès de `FLOW_TICK` événements qui fournissent des mesures plus complètes que `FLOW_TURN`.

`FLOW_TICK` fournit un moyen de vérifier périodiquement l'existence de certaines conditions sur le flux, telles que l'absence de fenêtre ou les délais de Nagle, puis de prendre une action, telle que le lancement d'une PCAP ou l'envoi d'un message syslog.

Voici un exemple de `FLOW_TICK`:

```
log("RTT " + Flow.roundTripTime);
Remote.Syslog.info(
  " eh_event=FLOW_TICK" +
  " ClientIP="+Flow.client.ipaddr+
  " ServerIP="+Flow.server.ipaddr+
  " ServerPort="+Flow.server.port+
  " ServerName="+Flow.server.device.dnsNames[0]+
  " RTT="+Flow.roundTripTime);
```

`FLOW_TURN`

S'exécute à chaque tour TCP ou UDP. Un tour représente un cycle complet d'un client transfert des données de demande suivi du transfert d'une réponse par un serveur.

`FLOW_TURN` expose également un Turn objet.

Points de terminaison

Le flux fait référence à une conversation entre deux points de terminaison via un protocole ; un point de terminaison peut être l'un des composants suivants :

- `client`
- `server`
- `sender`
- `receiver`

Les méthodes et propriétés décrites dans cette section sont appelées ou accessibles pour un point de terminaison spécifié sur le flux. Par exemple, pour accéder au `device` propriété d'un client HTTP, la syntaxe est `Flow.client.device`.

Le point de terminaison que vous spécifiez dépend des événements associés au déclencheur. Par exemple, le `ACTIVEMQ_MESSAGE` L'événement ne prend en charge que les points de terminaison de l'expéditeur et du destinataire. Le tableau suivant affiche la liste des événements qui peuvent être associés à un flux et les points de terminaison pris en charge pour chaque événement :

Événement	Client/Serveur	Expéditeur/récepteur
AAA_REQUEST	oui	oui

Événement	Client/Serveur	Expéditeur/récepteur
AAA_RESPONSE	oui	oui
AJP_REQUEST	oui	oui
AJP_RESPONSE	oui	oui
ACTIVEMQ_MESSAGE	non	oui
CIFS_REQUEST	oui	oui
CIFS_RESPONSE	oui	oui
DB_REQUEST	oui	oui
DB_RESPONSE	oui	oui
DHCP_REQUEST	oui	oui
DHCP_RESPONSE	oui	oui
DICOM_REQUEST	oui	oui
DICOM_RESPONSE	oui	oui
DNS_REQUEST	oui	oui
DNS_RESPONSE	oui	oui
FIX_REQUEST	oui	oui
FIX_RESPONSE	oui	oui
FLOW_CLASSIFY	oui	non
FLOW_DETACH	oui	non
FLOW_RECORD	oui	non
FLOW_TICK	oui	non
FLOW_TURN	oui	non
FTP_REQUEST	oui	oui
FTP_RESPONSE	oui	oui
HL7_REQUEST	oui	oui
HL7_RESPONSE	oui	oui
HTTP_REQUEST	oui	oui
HTTP_RESPONSE	oui	oui
IBMMQ_REQUEST	oui	oui
IBMMQ_RESPONSE	oui	oui
ICA_AUTH	oui	non
ICA_CLOSE	oui	non
ICA_OPEN	oui	non
ICA_TICK	oui	non
ICMP_MESSAGE	non	oui

Événement	Client/Serveur	Expéditeur/récepteur
KERBEROS_REQUEST	oui	oui
KERBEROS_RESPONSE	oui	oui
LDAP_REQUEST	oui	oui
LDAP_RESPONSE	oui	oui
MEMCACHE_REQUEST	oui	oui
MEMCACHE_RESPONSE	oui	oui
MOBUS_REQUEST	oui	oui
MODBUS_RESPONSE	oui	oui
MONGODB_REQUEST	oui	oui
MONGODB_RESPONSE	oui	oui
MSMQ_MESSAGE	non	oui
NFS_REQUEST	oui	oui
NFS_RESPONSE	oui	oui
POP3_REQUEST	oui	oui
POP3_RESPONSE	oui	oui
REDIS_REQUEST	oui	oui
REDIS_RESPONSE	oui	oui
RDP_CLOSE	oui	non
RDP_OPEN	oui	non
RDP_TICK	oui	non
RTCP_MESSAGE	non	oui
RTP_CLOSE	non	oui
RTP_OPEN	non	oui
RTP_TICK	non	oui
SCCP_MESSAGE	non	oui
SIP_REQUEST	oui	oui
SIP_RESPONSE	oui	oui
SMPP_REQUEST	oui	oui
SMPP_RESPONSE	oui	oui
SMTP_REQUEST	oui	oui
SMTP_RESPONSE	oui	oui
SSL_ALERT	oui	oui
SSL_CLOSE	oui	non
SSL_HEARTBEAT	oui	oui

Événement	Client/Serveur	Expéditeur/récepteur
SSL_OPEN	oui	non
SSL_PAYLOAD	oui	oui
SSL_RECORD	oui	oui
SSL_RENEGOTIATE	oui	non
TCP_CLOSE	oui	non
TCP_OPEN	oui	non
TCP_PAYLOAD	oui	oui
UDP_PAYLOAD	oui	oui
TELNET_MESSAGE	oui	oui
WEBSOCKET_OPEN	oui	non
WEBSOCKET_CLOSE	oui	non
WEBSOCKET_MESSAGE	oui	oui

Méthodes de terminaison

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur FLOW_RECORD événement. Les validations d'enregistrement ne sont pas prises en charge sur FLOW_CLASSIFY, FLOW_DETACH, FLOW_TICK, ou FLOW_TURN événements.

Sur un flux, le trafic se déplace dans chaque direction entre deux points terminaux. Le `commitRecord()` Cette méthode n'enregistre les détails du flux que dans une seule direction, par exemple du client au serveur. Pour enregistrer les détails de l' ensemble du flux, vous devez appeler `commitRecord()` deux fois, une fois pour chaque direction, et spécifiez le point de terminaison dans la syntaxe, par exemple `Flow.client.commitRecord()` et `Flow.server.commitRecord()`.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Pour afficher les propriétés par défaut validées pour l' objet d'enregistrement, consultez `record` propriété ci-dessous.

Propriétés des terminaux

`bytes:` ***Numéro***

Le nombre de L4 octets de charge utile transmis par un équipement. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `Flow.client.bytes` ou `Flow.receiver.bytes`.

Accès uniquement sur FLOW_TICK, FLOW_TURN, ou FLOW_RECORD événements ; sinon, une erreur se produira.

`customDevices:` ***Array***

Un ensemble de périphériques personnalisés dans le flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.customDevices` ou `Flow.receiver.customDevices`.

`device:` ***Appareil***

Le Device objet associé à un équipement. Spécifiez le rôle de l'équipement dans la syntaxe. Par exemple, pour accéder à l'adresse MAC du client équipement, spécifiez `Flow.client.device.hwaddr`.

`equals:` ***Booléen***

Effectue un test d'égalité entre Device objets.

`dscp:` ***Numéro***

Le nombre représentant la dernière valeur du point de code de services différenciés (DSCP) du paquet de flux.

Spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `Flow.client.dscp` ou `Flow.server.dscp`.

`dscpBytes:` ***Array***

Un tableau contenant le nombre de L2 octets pour une valeur de point de code de services différenciés (DSCP) spécifique transmise par un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `Flow.client.dscpBytes` ou `Flow.server.dscpBytes`.

La valeur est zéro pour chaque entrée qui ne contient aucun octet du DSCP spécifique depuis la dernière `FLOW_TICK` événement.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`dscpName1:` ***Corde***

Le nom associé à la valeur DSCP transmise par le périphérique 1 dans le flux. Le tableau suivant répertorie les noms DSCP les plus connus :

Numéro	Nom
8	CS1
10	AF11
12	AF12
14	AF13
16	CS2
18	AF21
20	AF22
22	AF23
24	CS3
26	AF31
28	AF32
30	AF33
32	CS4
34	AF41
36	AF42
38	AF43
40	CS5

Numéro	Nom
44	VA
46	EF
48	CS6
56	CS7

`dscpName2:` ***Corde***

Le nom associé à la valeur DSCP transmise par le périphérique 2 dans le flux. Le tableau suivant répertorie les noms DSCP les plus connus :

Numéro	Nom
8	CS1
10	AF11
12	AF12
14	AF13
16	CS2
18	AF21
20	AF22
22	AF23
24	CS3
26	AF31
28	AF32
30	AF33
32	CS4
34	AF41
36	AF42
38	AF43
40	CS5
44	VA
46	EF
48	CS6
56	CS7

`dscpPkts:` ***Array***

Un tableau contenant le nombre de L2 paquets pour une valeur de point de code de services différenciés (DSCP) donnée transmis par un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `Flow.client.dscpPkts` ou `Flow.server.dscpPkts`.

La valeur est zéro pour chaque entrée qui ne contient aucun paquet du DSCP spécifique depuis la dernière `FLOW_TICK` événement.

S'applique uniquement à FLOW_TICK ou FLOW_TURN événements.

`fragPkts:` ***Numéro***

Nombre de paquets résultant de la fragmentation IP transmis par un client ou un équipement serveur dans le flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `Flow.client.fragPkts` ou `Flow.server.fragPkts`.

Accès uniquement sur FLOW_TICK ou FLOW_TURN événements ; sinon, une erreur se produira.

`ipaddr1:` ***Adresse IP***

Le IPAddress objet associé à l'appareil 1 dans le flux.

`equals:` ***Booléen***

Effectue un test d'égalité entre IPAddress objets.

`ipaddr2:` ***Adresse IP***

Le IPAddress objet associé à l'appareil 2 dans le flux.

`equals:` ***Booléen***

Effectue un test d'égalité entre IPAddress objets.

`isAborted:` ***Booléen***

La valeur est `true` si un flux TCP a été interrompu par une réinitialisation TCP (RST). Le flux peut être interrompu par un équipement. Le cas échéant, spécifiez le rôle de l' équipement dans la syntaxe, par exemple, `Flow.client.isAborted` ou `Flow.receiver.isAborted`.

Cette condition peut être détectée dans TCP_CLOSE événement et quel que soit l'impact L7 événements (par exemple, HTTP_REQUEST ou DB_RESPONSE).

Note:
- Un L4 l'abandon se produit lorsqu'une connexion TCP est fermée par un RST au lieu d'un arrêt progressif.
- Un abandon de réponse L7 se produit lorsqu'une connexion se ferme au milieu d'une réponse. Cela peut être dû à un RST, à un arrêt progressif du FIN ou à une expiration.
- L'abandon d'une demande L7 se produit lorsqu'une connexion se ferme au milieu d'une demande. Cela peut également être dû à un RST, à un arrêt progressif du FIN ou à une expiration.

`isShutdown:` ***Booléen***

La valeur est `true` si l'équipement a initié l'arrêt de la connexion TCP. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `Flow.client.isShutdown` ou `Flow.receiver.isShutdown`.

`l2Bytes:` ***Numéro***

Le nombre de L2 octets, y compris les en-têtes Ethernet, transmis par un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.l2Bytes` ou `Flow.server.l2Bytes`.

Accès uniquement sur FLOW_TICK ou FLOW_TURN événements ; sinon, une erreur se produira.

`nagleDelay:` ***Numéro***

Le nombre de retards Nagle associés à un équipement dans le flux. Spécifiez le rôle de l' équipement dans la syntaxe, par exemple `Flow.client.nagleDelay` ou `Flow.server.nagleDelay`.

Accès uniquement sur FLOW_TICK ou FLOW_TURN événements ; sinon, une erreur se produira.

`overlapFragPkts:` ***Numéro***

Nombre de paquets de fragments IP non identiques dont les données se chevauchent et transmis par un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.overlapFragPkts` ou `Flow.server.overlapFragPkts`.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`overlapSegments:` ***Numéro***

Le nombre de segments TCP non identiques, transmis par un équipement du flux, où deux segments TCP ou plus contiennent des données pour la même partie du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.overlapSegments` ou `Flow.server.overlapSegments`.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`payload:` ***Tampon***

La charge utile Tampon associé à un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `Flow.client.payload` ou `Flow.receiver.payload`.

Accès uniquement sur `TCP_PAYLOAD`, `UDP_PAYLOAD`, ou `SSL_PAYLOAD` événements ; sinon, une erreur se produira.

`pkts:` ***Numéro***

Le nombre de paquets transmis par un équipement dans le flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.pkts` ou `Flow.server.pkts`.

Accès uniquement sur `FLOW_TICK`, `FLOW_TURN`, ou `FLOW_RECORD` événements ; dans le cas contraire, une erreur se produira.

`port:` ***Numéro***

Numéro de port associé à un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.port` ou `Flow.receiver.port`.

`rcvWndThrottle:` ***Numéro***

Le nombre de régulateurs de fenêtre de réception envoyés depuis un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.rcvWndThrottle` ou `Flow.server.rcvWndThrottle`.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `Flow.commitRecord()` sur un `FLOW_RECORD` événement. L'objet d'enregistrement représente les données provenant d'une seule direction du flux.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `age`
- `bytes (L3)`

  Note: Cette propriété représente le nombre total d'octets transmis par le flux au moment de l'exécution de l'événement FLOW_RECORD . L'événement FLOW_RECORD s'exécute plusieurs fois au cours de chaque flux, de sorte que la valeur augmente à chaque exécution de l'événement.
- `clientIsExternal`
- `dscpName`
- `first`
- `firstPayloadBytes`

Représentation hexadécimale des 16 premiers octets de charge utile du flux.

- `last`
- `pkts`
- `proto`
- `receiverAddr`
- `receiverIsExternal`
- `receiverPort`
- `roundTripTime`

  Le temps aller-retour (RTT) le plus récent de ce flux. Un RTT est le temps qu'il a fallu à un équipement pour envoyer un paquet et recevoir un accusé de réception immédiat (ACK).
- `senderAddr`
- `senderIsExternal`
- `senderPort`
- `serverIsExternal`
- `tcpFlags`

Spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `Flow.client.record` ou `Flow.server.record`.

Accédez à l'objet d'enregistrement uniquement sur `FLOW_RECORD` événements ; dans le cas contraire, une erreur se produira.

`rto:` ***Numéro***

Le nombre de délais de retransmission (RTO) associé à un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.rto` ou `Flow.server.rto`.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`totalL2Bytes`

Le nombre d'octets L2 envoyés par un équipement pendant le flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.totalL2Bytes` ou `Flow.server.totalL2Bytes`.

`totalL2Bytes1:` ***Numéro***

Le nombre d'octets L2 envoyés pendant le flux par le périphérique 1.

`totalL2Bytes2:` ***Numéro***

Le nombre d'octets L2 envoyés pendant le flux par le périphérique 2.

`zeroWnd:` ***Numéro***

Nombre de fenêtres nulles envoyées depuis un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.zeroWnd` ou `Flow.server.zeroWnd`.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

Méthodes

`addApplication(name:` ***Corde*** `, turnTiming:` ***Booléen*** `):` ***vide***

Crée une application portant le nom spécifié et collecte les métriques L2-L4 à partir du flux. L'application peut être consultée dans le système ExtraHop et les métriques sont affichées sur une page L4 de l'application. Un flux peut être associé à une ou plusieurs applications à un instant donné ; les métriques L2-L4 collectées par chaque application seront les mêmes.

Appel `Flow.addApplication(name)` sur un `FLOW_CLASSIFY` cet événement est courant sur les protocoles non pris en charge. Pour les flux sur les protocoles pris en charge avec L7 événements

déclencheurs, il est recommandé d'appeler le `Application(name).commit()` méthode, qui collecte un ensemble plus important de métriques de protocole.

L'optionnel `turnTiming` flag est défini sur false par défaut. S'il est défini sur true, le système ExtraHop collecte des mesures supplémentaires de chronométrage des tours pour le flux. Si cet indicateur est omis, aucune métrique de chronométrage des tours n'est enregistrée pour l'application sur le flux associé. Analyses d'analyse du temps de rotation L4 comportement afin de déduire les temps de traitement L7 lorsque le protocole surveillé suit un modèle de demande client et de réponse du serveur et dans lequel le client envoie le premier message. Les protocoles « bannières » (dans lesquels le serveur envoie le premier message) et les protocoles dans lesquels les données circulent simultanément dans les deux sens ne sont pas recommandés pour l'analyse de la synchronisation des tours.

`captureStart(name:` ***Corde*** `, options:` ***Objet*** `):` ***Corde***

Lance une capture de paquets de précision (PPCAP) pour le flux et renvoie un identifiant unique de la capture de paquets sous la forme d'un nombre décimal sous forme de chaîne. Retours `null` si la PCAP ne démarre pas.

`name:` ***Corde***

Le nom du fichier de capture de paquets.

- La longueur maximale est de 256 caractères
- Une capture distincte est créée pour chaque flux.
- Les fichiers de capture portant le même nom sont différenciés par des horodatages.

`options:` ***Objet***

Les options contenues dans l'objet de capture. Omettez l'une des options pour indiquer une taille illimitée pour cette option. Toutes les options s'appliquent à l'ensemble du flux, à l'exception des options « rétrospectives » qui s'appliquent uniquement à la partie du flux précédant l'événement déclencheur qui a lancé la capture de paquets.

`maxBytes:` ***Numéro***

Le nombre maximum total d'octets.

`maxBytesLookback:` ***Numéro***

Le nombre maximum total d'octets provenant de la mémoire tampon de visualisation. Le tampon de retour fait référence aux paquets capturés avant l'appel à `Flow.captureStart()`.

`maxDurationMSec:` ***Numéro***

Durée maximale de la PCAP, exprimée en millisecondes.

`maxPackets:` ***Numéro***

Le nombre maximum total de paquets. La valeur maximale peut être dépassée si charge du déclencheur est lourd.

`maxPacketsLookback:` ***Numéro***

Le nombre maximum de paquets provenant de la mémoire tampon de visualisation. Le tampon de retour fait référence aux paquets capturés avant l'appel à `Flow.captureStart()`.

Voici un exemple de `Flow.captureStart()`:

```
// EVENT: HTTP_REQUEST
// capture facebook HTTP traffic flows
if (HTTP.uri.indexOf("www.facebook.com") !== -1) {
   var name = "facebook-" + HTTP.uri;
   //packet capture options: capture 20 packets, up to 10 from the
 lookback buffer
   var opts = {
      maxPackets: 20,
      maxPacketsLookback: 10
```

```
    };
    Flow.captureStart(name, opts);
}
```

Note:
- Le `Flow.captureStart()` L'appel de fonction nécessite que vous disposiez d'une licence pour la capture de paquets de précision.
- Vous pouvez spécifier le nombre d'octets par paquet (snaplen) que vous souhaitez capturer lors de la configuration du déclencheur dans le système ExtraHop. Cette option n'est disponible que pour certains événements. Voir Options de déclencheur avancées pour plus d' informations.
- Sur les systèmes ExtraHop Performance, les fichiers capturés sont disponibles dans les paramètres d' administration. Sur les systèmes RevealX, les fichiers capturés sont disponibles depuis la page Paquets du système ExtraHop.
- Sur les systèmes ExtraHop Performance, si le disque de capture de paquets de précision est plein, aucune nouvelle capture n'est enregistrée tant que l'utilisateur n'a pas supprimé les fichiers manuellement. Sur les systèmes Reveal, les anciennes captures de paquets sont supprimées lorsque le disque de capture de paquets de précision est plein pour permettre au système de continuer à enregistrer de nouvelles captures de paquets.
- La longueur maximale de la chaîne de nom de fichier est de 256 caractères. Si le nom dépasse 256 caractères, il sera tronqué et un message d'avertissement sera visible dans le journal de débogage, mais le déclencheur continuera à s'exécuter.
- La taille du fichier de capture est la valeur maximale atteinte en premier entre `maxPackets` et `maxBytes` options.
- La taille de la mémoire tampon de capture est la valeur maximale atteinte en premier entre `maxPacketsLookback` et `maxBytesLookback` options.
- Chacun a réussi `max*` le paramètre capturera jusqu'à la limite de paquet suivante.
- Si la PCAP a déjà été lancée sur le flux en cours, `Flow.captureStart()` les appels génèrent un avertissement visible dans le journal de débogage, mais le déclencheur continuera à fonctionner.
- Il existe un maximum de 128 captures de paquets simultanées dans le système. Si cette limite est atteinte, les appels suivants à `Flow.captureStart()` générera un avertissement visible dans le journal de débogage, mais le déclencheur continuera à s'exécuter.

`captureStop()`: ***Booléen***

Arrête une PCAP en cours sur le flux actuel.

`commitRecord1()`: ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré qui représente les données envoyées depuis `device1` dans une seule direction sur le flux.

Vous ne pouvez appeler cette méthode que sur `FLOW_RECORD` événements, et chaque enregistrement unique n'est validé qu'une seule fois pour les enregistrements intégrés.

Pour afficher les propriétés validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

`commitRecord2()`: ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré qui représente les données envoyées depuis `device2` dans une seule direction sur le flux.

Vous ne pouvez appeler cette méthode que sur `FLOW_RECORD` événements, et chaque enregistrement unique n'est validé qu'une seule fois pour les enregistrements intégrés.

Pour afficher les propriétés validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

`findCustomDevice(deviceID:` ***Corde*** `):` ***Appareil***

Renvoie un seul Device objet qui correspond au paramètre DeviceID spécifié si l'équipement est situé de part et d'autre du flux. Retours `null` si aucun équipement correspondant n'est trouvé.

`getApplications():` ***Corde***

Récupère toutes les applications associées au flux.

Propriétés

Les propriétés et méthodes de l'objet Flow décrites dans cette section sont accessibles à tous L7 événement déclencheur associé au flux.

Par défaut, le système ExtraHop utilise une classification des protocoles mal initiée. Il essaiera donc de classer les flux même après le lancement de la connexion. L'initiation libre peut être désactivée pour les ports qui ne transportent pas toujours le trafic du protocole (par exemple, le port générique 0). Pour de tels flux, `device1`, `port1`, et `ipaddr1` représenter l'équipement dont l'adresse IP est numériquement inférieure et `device2`, `port2`, et `ipaddr2` représente l'équipement dont l'adresse IP est numériquement la plus élevée.

`age:` ***Numéro***

Le temps écoulé depuis le début du flux, exprimé en secondes.

`bytes1:` ***Numéro***

Le nombre de L4 octets de charge utile transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `bytes2`. L' équipement représenté par `bytes1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK`, `FLOW_TURN`, ou `FLOW_RECORD` événements ; sinon, une erreur se produira.

`bytes2:` ***Numéro***

Le nombre de L4 octets de charge utile transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `bytes1`. L' équipement représenté par `bytes2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK`, `FLOW_TURN`, ou `FLOW_RECORD` événements ; sinon, une erreur se produira.

`customDevices1:` ***Array***

Une gamme de produits personnalisés Device objets d'un flux. Les appareils personnalisés situés de l'autre côté du flux sont disponibles en accédant `customDevices2`. L'équipement représenté par `customDevices1` reste constant pour le flux.

`customDevices2:` ***Array***

Une gamme de produits personnalisés Device objets d'un flux. Les appareils personnalisés situés de l'autre côté du flux sont disponibles en accédant `customDevices1`. L'équipement représenté par `customDevices2` reste constant pour le flux.

`device1:` ***Appareil***

Le Device objet associé à l'un des deux appareils du flux ; l'autre équipement est représenté par `device2`. L' équipement représenté par `device1` reste constant pour le flux. Par exemple, `Flow.device1.hwaddr` accède aux adresses MAC de cet équipement dans le flux.

`equals:` ***Booléen***

Effectue un test d'égalité entre Device objets.

`device2:` ***Appareil***

Le Device objet associé à l'un des deux appareils du flux ; l'autre équipement est représenté par `device1`. L' équipement représenté par `device2` reste constant pour le flux. Par exemple, `Flow.device2.hwaddr` accède aux adresses MAC de cet équipement dans le flux.

`equals:` ***Booléen***

Effectue un test d'égalité entre Device objets.

`dscp1:` ***Numéro***

Le numéro représentant la dernière valeur de point de code de services différenciés (DSCP) transmise par l'un des deux appareils du flux ; l'autre équipement est représenté par `dscp2`. L'équipement représenté par `dscp1` reste constant pour le flux.

`dscp2:` ***Numéro***

Le lnumber représentant la dernière valeur de point de code de services différenciés (DSCP) transmise par l'un des deux appareils du flux ; l'autre équipement est représenté par `dscp1`. L'équipement représenté par `dscp2` reste constant pour le flux.

`dscpBytes1:` ***Array***

Un tableau contenant le nombre de L2 octets pour une valeur de point de code de services différenciés (DSCP) spécifique transmise par l'un des deux périphériques du flux ; l'autre équipement est représenté par `dscpBytes2`. L'équipement représenté par `dscpBytes1` reste constant pour le flux.

La valeur est zéro pour chaque entrée qui ne contient aucun octet du DSCP spécifique depuis la dernière `FLOW_TICK` événement.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`dscpBytes2:` ***Array***

Un tableau contenant le nombre de L2 octets pour une valeur de point de code de services différenciés (DSCP) spécifique transmise par l'un des deux périphériques du flux ; l'autre équipement est représenté par `dscpBytes1`. L'équipement représenté par `dscpBytes2` reste constant pour le flux.

La valeur est zéro pour chaque entrée qui ne contient aucun octet du DSCP spécifique depuis la dernière `FLOW_TICK` événement.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`dscpName1:` ***Corde***

Le nom associé à la valeur DSCP transmise par l'un des deux périphériques du flux ; l'autre équipement est représenté par `dscpName2`. L'équipement représenté par `dscpName1` reste constant pour le flux.

Consultez les `dscpName` propriété dans le Points de terminaison section pour une liste des noms de code DSCP pris en charge.

`dscpName2:` ***Corde***

Le nom associé à la valeur DSCP transmise par l'un des deux périphériques du flux ; l'autre équipement est représenté par `dscpName1`. L'équipement représenté par `dscpName2` reste constant pour le flux.

Consultez les `dscpName` propriété dans le Points de terminaison section pour une liste des noms de code DSCP pris en charge.

`dscpPkts1:` ***Array***

Un tableau contenant le nombre de L2 paquets pour une valeur de point de code de services différenciés (DSCP) donnée transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `dscpPkts2`. L'équipement représenté par `dscpPkts1` reste constant pour le flux.

La valeur est zéro pour chaque entrée qui ne contient aucun paquet du DSCP spécifique depuis la dernière `FLOW_TICK` événement.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`dscpPkts2:` ***Array***

Un tableau contenant le nombre de L2 paquets pour une valeur de point de code de services différenciés (DSCP) donnée transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `dscpPkts1`. L'équipement représenté par `dscpPkts2` reste constant pour le flux.

La valeur est zéro pour chaque entrée qui ne contient aucun paquet du DSCP spécifique depuis la dernière `FLOW_TICK` événement.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`fragPkts1:` ***Numéro***

Le nombre de paquets résultant de la fragmentation IP transmis par l'un des deux appareils du flux ; l'autre équipement est représenté par `fragPkts2`. L' équipement représenté par `fragPkts1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`fragPkts2:` ***Numéro***

Le nombre de paquets résultant de la fragmentation IP transmis par l'un des deux appareils du flux ; l'autre équipement est représenté par `fragPkts1`. L' équipement représenté par `fragPkts2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`id:` ***Corde***

L'identifiant unique d'un enregistrement Flow.

`ipaddr:` ***Adresse IP***

Le IPAddress objet associé à un équipement du flux. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.ipaddr` ou `Flow.receiver.ipaddr`.

`equals:` ***Booléen***

Effectue un test d'égalité entre IPAddress objets.

`ipproto:` ***Corde***

Le protocole IP associé au flux, tel que TCP ou UDP.

`ipver:` ***Corde***

Version IP associée au flux, telle que IPv4 ou IPv6.

`isAborted:` ***Booléen***

La valeur est `true` si un flux TCP a été interrompu par une réinitialisation TCP (RST). Le flux peut être interrompu par un équipement. Le cas échéant, spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `Flow.client.isAborted` ou `Flow.receiver.isAborted`.

Cette condition peut être détectée dans `TCP_CLOSE` événement et quel que soit l'impact L7 événements (par exemple, `HTTP_REQUEST` ou `DB_RESPONSE`).

Note:
- Un L4 l'abandon se produit lorsqu'une connexion TCP est fermée par un RST au lieu d'un arrêt progressif.
- Un abandon de réponse L7 se produit lorsqu'une connexion se ferme alors qu'une réponse est en cours de réponse. Cela peut être dû à un RST, à un arrêt progressif du FIN ou à une expiration.
- L'abandon d'une demande L7 se produit lorsqu'une connexion se ferme au milieu d'une demande. Cela peut également être dû à un RST, à un arrêt progressif du FIN ou à une expiration.

`isExpired:` ***Booléen***

La valeur est `true` si le flux a expiré au moment de l'événement.

`isShutdown:` ***Booléen***

La valeur est `true` si l'équipement a initié l'arrêt de la connexion TCP. Spécifiez le rôle de l'équipement dans la syntaxe, par exemple `Flow.client.isShutdown` ou `Flow.receiver.isShutdown`.

`l2Bytes1:` ***Numéro***

Le nombre de L2 octets, y compris les en-têtes Ethernet, transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `l2Bytes2`. L'équipement représenté par `l2Bytes1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`l2Bytes2:` ***Numéro***

Le nombre de L2 octets, y compris les en-têtes Ethernet, transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `l2Bytes1`. L'équipement représenté par `l2Bytes2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`l7proto:` ***Corde***

Le protocole L7 associé au flux. Pour les protocoles connus, la propriété renvoie une chaîne représentant le nom du protocole, tel que HTTP, DHCP, Memcache. Pour les protocoles moins connus, la propriété renvoie une chaîne au format `ipproto:port—tcp:13724` ou `udp:11258` Pour les noms de protocoles personnalisés, la propriété renvoie une chaîne représentant le nom défini dans la section Classification des protocoles des paramètres d' administration.

Cette propriété n'est pas valide pendant `TCP_OPEN` événements.

`nagleDelay1:` ***Numéro***

Le nombre de retards de Nagle associés à l'un des deux appareils du flux ; l'autre équipement est représenté par `nagleDelay2`. L'équipement représenté par `nagleDelay1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`nagleDelay2:` ***Numéro***

Le nombre de retards de Nagle associés à l'un des deux appareils du flux ; l'autre équipement est représenté par `nagleDelay1`. L'équipement représenté par `nagleDelay2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`overlapFragPkts1:` ***Numéro***

Le nombre de paquets de fragments IP non identiques transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `overlapFragPkts2`. L' équipement représenté par `overlapFragPkts1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`overlapFragPkts2:` ***Numéro***

Le nombre de paquets de fragments IP non identiques transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `overlapFragPkts1`. L' équipement représenté par `overlapFragPkts2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`overlapSegments1:` ***Numéro***

Le nombre de segments TCP non identiques dans lesquels deux segments ou plus contiennent des données pour la même partie du flux. Les segments TCP sont transmis par l'un des deux équipements du flux ; l'autre équipement est représenté par `overlapSegments2`. L'équipement représenté par `overlapSegments1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`overlapSegments2:` ***Numéro***

Le nombre de segments TCP non identiques dans lesquels deux segments ou plus contiennent des données pour la même partie du flux. Les segments TCP sont transmis par l'un des deux équipements du flux ; l'autre équipement est représenté par `overlapSegments1`. L'équipement représenté par `overlapSegments2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`payload1:` ***Tampon***

La charge utile Tampon associé à l'un des deux appareils du flux ; l'autre équipement est représenté par `payload2`. L' équipement représenté par `payload1` reste constant pour le flux.

Accès uniquement sur `TCP_PAYLOAD`, `UDP_PAYLOAD`, et `SSL_PAYLOAD` événements ; sinon, une erreur se produira.

`payload2:` ***Tampon***

La charge utile Tampon associé à l'un des deux appareils du flux ; l'autre équipement est représenté par `payload1`. L' équipement représenté par `payload2` reste constant pour le flux.

Accès uniquement sur `TCP_PAYLOAD`, `UDP_PAYLOAD`, ou `SSL_PAYLOAD` événements ; sinon, une erreur se produira.

`pkts1:` ***Numéro***

Le nombre de paquets transmis par l'un des deux appareils du flux ; l'autre équipement est représenté par `pkts2`. L'équipement représenté par `pkts1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK`, `FLOW_TURN`, ou `FLOW_RECORD` événements ; sinon, une erreur se produira.

`pkts2:` ***Numéro***

Le nombre de paquets transmis par l'un des deux appareils du flux ; l'autre équipement est représenté par `pkts1`. L'équipement représenté par `pkts2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK`, `FLOW_TURN`, ou `FLOW_RECORD` événements ; sinon, une erreur se produira.

`port1:` ***Numéro***

Le numéro de port associé à l'un des deux périphériques d'un flux ; l'autre appareil est représenté par `port2`. L'équipement représenté par `port1` reste constant pour le flux.

`port2:` ***Numéro***

Le numéro de port associé à l'un des deux périphériques d'un flux ; l'autre appareil est représenté par `port1`. L'équipement représenté par `port2` reste constant pour le flux.

`rcvWndThrottle1:` ***Numéro***

Le nombre de régulateurs de fenêtre de réception envoyés par l'un des deux appareils du flux ; l'autre appareil est représenté par `rcvWndThrottle2`. L'équipement représenté par `rcvWndThrottle1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`rcvWndThrottle2:` ***Numéro***

Le nombre de régulateurs de fenêtre de réception envoyés par l'un des deux appareils du flux ; l'autre appareil est représenté par `rcvWndThrottle1`. L'équipement représenté par `rcvWndThrottle2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`record1:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `Flow.commitRecord1()` sur un `FLOW_RECORD` événement.

L' objet représente le trafic envoyé dans une seule direction depuis l'un des deux appareils du flux ; l'autre appareil est représenté par le `record2` propriété. L' équipement représenté par `record1` la propriété reste constante pour le flux.

Accédez à l'objet d'enregistrement uniquement sur `FLOW_RECORD` événements ; sinon, une erreur se produira.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `age`
- `bytes (L3)`
- `clientIsExternal`
- `dscpName`
- `first`
- `last`
- `pkts`
- `proto`
- `receiverAddr`
- `receiverIsExternal`
- `receiverPort`
- `roundTripTime`

  Le temps aller-retour (RTT) le plus récent de ce flux. Un RTT est le temps qu'il a fallu à un équipement pour envoyer un paquet et recevoir un accusé de réception immédiat (ACK).
- `senderAddr`
- `senderIsExternal`
- `senderPort`
- `serverIsExternal`
- `tcpOrigin`

  Ce champ d'enregistrement n'est inclus que si l'enregistrement représente le trafic envoyé depuis un équipement client ou expéditeur.
- `tcpFlags`

`record2:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `Flow.commitRecord2()` sur un `FLOW_RECORD` événement.

L' objet représente le trafic envoyé dans une seule direction depuis l'un des deux appareils du flux ; l'autre appareil est représenté par le `record1` propriété. L' équipement représenté par `record2` la propriété reste constante pour le flux.

Accédez à l'objet d'enregistrement uniquement sur `FLOW_RECORD` événements ; sinon, une erreur se produira.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `age`
- `bytes (L3)`
- `clientIsExternal`
- `dscpName`
- `first`
- `last`
- `pkts`
- `proto`
- `receiverAddr`
- `receiverIsExternal`
- `receiverPort`

- `roundTripTime`

  Le temps aller-retour (RTT) le plus récent de ce flux. Un RTT est le temps qu'il a fallu à un équipement pour envoyer un paquet et recevoir un accusé de réception immédiat (ACK).
- `senderAddr`
- `senderIsExternal`
- `senderPort`
- `serverIsExternal`
- `tcpOrigin`

  Ce champ d'enregistrement n'est inclus que si l'enregistrement représente le trafic envoyé depuis un équipement client ou expéditeur.
- `tcpFlags`

`roundTripTime:` ***Numéro***

Temps médian aller-retour (RTT) pendant la durée de l'événement, exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`rto1:` ***Numéro***

Le nombre de délais de retransmission (RTO) associé à l'un des deux appareils du flux ; l'autre appareil est représenté par `rto2`. L'équipement représenté par `rto1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`rto2:` ***Numéro***

Le nombre de délais de retransmission (RTO) associé à l'un des deux appareils du flux ; l'autre appareil est représenté par `rto1`. L'équipement représenté par `rto2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`store:` ***Objet***

Le magasin de flux est conçu pour transmettre des objets de la demande à la réponse sur le même flux. Le `store` object est une instance d'un objet JavaScript vide. Les objets peuvent être attachés au magasin en tant que propriétés en définissant la clé de propriété et la valeur de la propriété. Par exemple :

```
Flow.store.myobject = "myvalue";
```

Pour les événements qui se produisent sur le même flux, vous pouvez appliquer le magasin de flux au lieu de la table de session pour partager des informations. Par exemple :

```
// request
Flow.store.userAgent = HTTP.userAgent;

// response
var userAgent = Flow.store.userAgent;
```

Important: Les valeurs du magasin de flux persistent pour toutes les demandes et réponses transmises par ce flux. Lorsque vous utilisez le magasin de flux, il est recommandé de définir la variable de stockage de flux sur `null` lorsque sa valeur ne doit pas être transmise à la demande ou à la réponse suivante. Cette pratique présente l'avantage supplémentaire de conserver la mémoire de stockage des flux.

La plupart des déclencheurs de stockage de flux doivent avoir une structure similaire à l' exemple suivant :

```
if (event === 'DB_REQUEST') {
                if (DB.statement) {
```

```
                Flow.store.stmt = DB.statement;
} else {
                Flow.store.stmt = null;
}
}
else if (event === 'DB_RESPONSE') {
        var stmt = Flow.store.stmt;
        Flow.store.stmt = null;
        if (stmt) {
                // Do something with 'stmt';
                // for example, commit a metric
        }
}
```

Note: Étant donné que les demandes DHCP se produisent souvent sur des flux différents des réponses DHCP correspondantes, nous vous recommandons de combiner les informations de demande et de réponse DHCP en stockant les ID de transaction DHCP dans la table de session. Par exemple, le code de déclencheur suivant crée une métrique qui permet de suivre le nombre de messages de découverte DHCP qui ont reçu un message d'offre DHCP correspondant :

```
if (event === 'DHCP_REQUEST'){
    var opts = {
        expire: 30
    };
    Session.add(DHCP.txId.toString(), DHCP.msgType, opts);
}
else if (event === 'DHCP_RESPONSE'){
    var reqMsgType = Session.lookup(DHCP.txId.toString());
    if (reqMsgType && DHCP.msgType === 'DHCPOFFER') {
        Device.metricAddCount('dhcp-discover-offer', 1);
    }
}
```

`tcpOrigin:` ***Adresse IP | Null***

L'adresse IP d'origine du client ou de l'expéditeur si elle est spécifiée par un proxy réseau dans l'option TCP 28.

`vlan:` ***Numéro***

Le numéro de VLAN associé au flux. Si aucune balise VLAN n'est présente, cette valeur est définie sur `0`.

`vxlanVNI:` ***Numéro***

Numéro d'identifiant réseau VXLAN associé au flux. Si aucune balise VXLAN n'est présente, cette valeur est définie sur `NaN` .

`zeroWnd1:` ***Numéro***

Le nombre de fenêtres nulles associées à l'un des deux appareils du flux ; l'autre équipement est représenté par `zeroWnd2`. L'équipement représenté par `zeroWnd1` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

`zeroWnd2:` ***Numéro***

Le nombre de fenêtres nulles associées à l'un des deux appareils du flux ; l'autre équipement est représenté par `zeroWnd1`. L'équipement représenté par `zeroWnd2` reste constant pour le flux.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; sinon, une erreur se produira.

Exemples de déclencheurs

- Exemple : Surveiller les actions des PME sur les appareils
- Exemple : suivi des réponses HTTP de niveau 500 par ID client et URI

- Exemple : analyse de messages PoS personnalisés avec une analyse de charge utile universelle
- Exemple : analyse du syslog sur TCP avec une analyse de charge utile universelle
- Exemple : analyse NTP avec analyse de charge utile universelle
- Exemple : suivre les requêtes SOAP

`FlowInterface`

Le `FlowInterface` La classe vous permet de récupérer les attributs de l'interface de flux et d'ajouter des métriques personnalisées au niveau de l'interface.

Méthodes

`FlowInterface(id:` ***chaîne*** `)`

Constructeur de l'objet FlowInterface qui accepte un ID d'interface de flux. Une erreur se produit si l'identifiant de l'interface de flux n'existe pas sur le système ExtraHop.

Méthodes d'instance

Les méthodes décrites dans cette section vous permettent de créer des mesures personnalisées sur une interface de flux. Les méthodes ne sont présentes que sur les instances de NetFlow classe. Par exemple, l'instruction suivante collecte des métriques du trafic NetFlow sur l'interface d' entrée :

```
NetFlow.ingressInterface.metricAddCount("slow_rsp", 1);
```

Cependant, vous pouvez appeler la méthode FlowInterface en tant que méthode statique sur `NETFLOW_RECORD` événements. Par exemple, l'instruction suivante collecte des métriques du trafic NetFlow sur les interfaces d'entrée et de sortie :

```
FlowInterface.metricAddCount("slow_rsp", 1);
```

`metricAddCount(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de comptage. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Le nom de la métrique de comptage de niveau supérieur.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailCount(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de comptage par lequel vous pouvez approfondir. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDataset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de l'ensemble de données. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Nom de la métrique du jeu de données de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailDataset(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de l'ensemble de données par lequel vous pouvez approfondir. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDistinct(metric_name:` ***Corde*** `, item:` ***Numéro*** `|` ***Corde*** `|` ***Adresse IP*** `:void`

Crée une personnalisation niveau supérieur métrique de comptage distincte. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Nom de la métrique de comptage distincte de niveau supérieur.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

`metricAddDetailDistinct(metric_name:` ***Corde*** `, key:` ***Corde*** | ***Adresse IP*** `, item:` ***Numéro*** | ***Corde*** | ***Adresse IP*** `:void`

Crée une personnalisation détail métrique de comptage distincte par lequel vous pouvez approfondir. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Nom de la métrique de comptage distincte détaillée.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

`metricAddMax(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique maximale. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Le nom de la métrique maximale de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailMax(metric_name:` ***Corde*** `, key:` ***Corde*** | ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique maximale par lequel vous pouvez approfondir. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Nom de la métrique maximale de détail.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSampleset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur Sampleset métrique. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Le nom de la métrique de l'ensemble d'échantillons de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSampleset(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail Sampleset métrique par lequel vous pouvez approfondir. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSnap(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de capture d'écran. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Nom de la métrique de capture instantanée de niveau supérieur.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSnap(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de capture d'écran par lequel vous pouvez approfondir. Valide les données métriques dans l'interface de flux spécifiée.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

Propriétés de l'instance

`id:` ***Corde***

Chaîne qui identifie de manière unique l'interface de flux.

`number:` ***Numéro***

Numéro d'interface de flux indiqué par l'enregistrement NetFlow.

`FlowNetwork`

Le `FlowNetwork` class vous permet de récupérer les attributs du réseau de flux et d'ajouter des métriques personnalisées au niveau du réseau de flux.

Méthodes

`FlowNetwork(id:` ***chaîne*** `)`

Un constructeur pour l'objet FlowNetwork qui accepte un identifiant de réseau de flux. Une erreur se produit si l'identifiant du réseau de flux n'existe pas sur le système ExtraHop.

Méthodes d'instance

Les méthodes décrites dans cette section vous permettent de créer des mesures personnalisées sur un réseau de flux. Les méthodes ne sont présentes que sur les instances de NetFlow classe. Par exemple, l'instruction suivante collecte des métriques du trafic NetFlow sur un réseau individuel :

```
NetFlow.network.metricAddCount("slow_rsp", 1);
```

Cependant, vous pouvez appeler la méthode FlowNetwork en tant que méthode statique sur `NETFLOW_RECORD` événements. Par exemple, l'instruction suivante collecte des mesures du trafic NetFlow sur les deux appareils du réseau de flux :

```
FlowNetwork.metricAddCount("slow_rsp", 1);
```

`metricAddCount(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de comptage. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Le nom de la métrique de comptage de niveau supérieur.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailCount(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de comptage par lequel vous pouvez approfondir. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde | Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDataset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de l'ensemble de données. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Nom de la métrique du jeu de données de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailDataset(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de l'ensemble de données par lequel vous pouvez approfondir. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde | Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDistinct(metric_name:` ***Corde*** `, item:` ***Numéro*** | ***Corde*** | ***Adresse IP*** `:void`

Crée une personnalisation niveau supérieur métrique de comptage distincte. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Nom de la métrique de comptage distincte de niveau supérieur.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

`metricAddDetailDistinct(metric_name:` ***Corde*** `, key:` ***Corde*** | ***Adresse IP*** `, item:` ***Numéro*** | ***Corde*** | ***Adresse IP*** `:void`

Crée une personnalisation détail métrique de comptage distincte par lequel vous pouvez approfondir. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Nom de la métrique de comptage distincte détaillée.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

`metricAddMax(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique maximale. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Le nom de la métrique maximale de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailMax(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique maximale par lequel vous pouvez approfondir. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Nom de la métrique maximale de détail.

`key:` ***Corde | Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSampleset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur Sampleset métrique. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Le nom de la métrique de l'ensemble d'échantillons de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSampleset(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail Sampleset métrique par lequel vous pouvez approfondir. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSnap(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de capture d'écran. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Nom de la métrique de capture instantanée de niveau supérieur.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSnap(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de capture d'écran par lequel vous pouvez approfondir. Valide les données métriques dans le réseau de flux spécifié.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

Propriétés de l'instance

`id:` ***Corde***

Chaîne qui identifie de manière unique le réseau de flux.

`ipaddr:` ***Adresse IP***

L'adresse IP de l'interface de management sur le réseau de flux.

`GeoIP`

Le `GeoIP` class vous permet de récupérer la localisation approximative au niveau du pays ou de la ville d'une adresse spécifique.

Méthodes

Les valeurs renvoyées par les méthodes GeoIP sont obtenues à partir du Bases de données de pays ou de villes MaxMind GeoLite2 sauf configuration contraire par Source de données Geomap paramètres dans les paramètres d'administration.

À partir des paramètres de la source de données Geomap, vous pouvez télécharger des bases de données personnalisées et spécifier la base de données à référencer par défaut pour les recherches de villes ou de pays.

Nous vous recommandons de télécharger uniquement une base de données personnalisée au niveau de la ville si vous avez l'intention d'appeler les deux `GeoIP.getCountry()` et `GeoIP.getPreciseLocation()` méthodes dans les déclencheurs. Si les deux types de bases de données personnalisées sont chargés, le système ExtraHop extrait les valeurs des deux méthodes à partir de la base de données au niveau de la ville et ignore la base de données au niveau du pays, qui est considérée comme un sous-ensemble de la base de données au niveau de la ville.

`getCountry(ipaddr:` ***Adresse IP*** `):` ***Objet***

Renvoie les détails au niveau du pays pour la zone spécifiée IPAddress dans un objet contenant les champs suivants :

`continentName:` ***Corde***

Le nom du continent, tel que `Europe`, qui est associé au pays dont provient l'adresse IP spécifiée. La valeur est la même que `continentName` champ renvoyé par `getPreciseLocation()` méthode.

`continentCode:` ***Numéro***

Le code du continent, tel que `EU`, qui est associée à la valeur de `countryCode` champ, conformément à la norme ISO 3166. La valeur est la même que `continentCode` champ renvoyé par `getPreciseLocation()` méthode.

`countryName:` ***Corde***

Le nom du pays d'où provient l'adresse IP spécifiée, tel que `United States`. La valeur est la même que `countryName` champ renvoyé par `getPreciseLocation()` méthode.

`countryCode:` ***Corde***

Le code associé au pays, selon la norme ISO 3166, tel que `US`. La valeur est la même que `countryCode` champ renvoyé par `getPreciseLocation()` méthode.

Retours `null` dans un champ pour lequel aucune donnée n'est disponible, ou renvoie une `null` objet si toutes les données de champ ne sont pas disponibles.

Note: Le `getCountry()` La méthode nécessite 20 Mo de RAM totale sur le système ExtraHop, ce qui peut affecter les performances du système. La première fois que cette méthode est appelée dans un déclencheur, le système ExtraHop réserve la quantité de RAM requise, sauf si `getPreciseLocation()` la méthode a déjà été appelée. Le `getPreciseLocation()` la méthode nécessite 100 Mo de RAM, donc une quantité de RAM suffisante sera déjà disponible pour appeler le `getCountry()` méthode. La quantité de RAM requise n'est pas par déclencheur ou par appel de méthode ; le système ExtraHop ne réserve la quantité de RAM requise qu'une seule fois.

Dans l'exemple de code suivant , `getCountry()` La méthode est appelée pour chaque événement spécifié et récupère les données de localisation approximatives pour chaque adresse IP du client :

```
// ignore if the IP address is non-routable
if (Flow.client.ipaddr.isRFC1918) return;
var results=GeoIP.getCountry(Flow.client.ipaddr);
if (results) {
    countryCode=results.countryCode;
    // log the 2-letter country code of each IP address
    debug ("Country Code is " + results.countryCode);
}
```

`getPreciseLocation(ipaddr:` ***Adresse IP*** `):` ***Objet***

Renvoie les détails au niveau de la ville pour la zone spécifiée IPAddress dans un objet contenant les champs suivants :

`continentName:` ***Corde***

Le nom du continent, tel que `Europe`, qui est associé au pays dont provient l'adresse IP spécifiée. La valeur est la même que `continentName` champ renvoyé par `getCountry()` méthode.

`continentCode:` ***Numéro***

Le code du continent, tel que `EU`, qui est associée à la valeur de `countryCode` champ, conformément à la norme ISO 3166. La valeur est la même que `continentCode` champ renvoyé par `getCountry()` méthode.

`countryName:` ***Corde***

Le nom du pays d'où provient l'adresse IP spécifiée, tel que `United States`. La valeur est la même que `countryName` champ renvoyé par `getCountry()` méthode.

`countryCode:` ***Corde***

Le code associé au pays, selon la norme ISO 3166, tel que `US`. La valeur est la même que `countryCode` champ renvoyé par `getCountry()` méthode.

`region:` ***Corde***

La région, telle qu'un État ou une province, telle que `Washington`.

`city:` ***Corde***

La ville d'où provient l'adresse IP, telle que `Seattle`.

`latitude:` ***Numéro***

Latitude de l'emplacement de l'adresse IP.

`longitude:` ***Numéro***

Longitude de l'emplacement de l'adresse IP.

`radius:` ***Numéro***

Rayon, exprimé en kilomètres, autour des coordonnées de longitude et de latitude de l'emplacement de l'adresse IP.

Retours `null` dans un champ pour lequel aucune donnée n'est disponible, ou renvoie une `null` objet si toutes les données de champ ne sont pas disponibles.

Note: Le `getPreciseLocation()` La méthode nécessite 100 Mo de RAM totale sur le système ExtraHop, ce qui peut affecter les performances du système. La première fois que cette méthode est appelée dans un déclencheur, le système ExtraHop réserve la quantité de RAM requise, sauf si `getCountry()` la méthode a déjà été appelée. Le `getCountry()` La méthode nécessite 20 Mo de RAM, le système ExtraHop réserve donc 80 Mo de RAM supplémentaires. La quantité de RAM requise n'est pas par déclencheur ou par appel de méthode ; le système ExtraHop ne réserve la quantité de RAM requise qu'une seule fois.

`IPAddress`

Le `IPAddress` La classe vous permet de récupérer les attributs de l'adresse IP. La classe IPAddress est également disponible en tant que propriété pour la classe Flow.

Méthodes

`IPAddress(ip:` ***Corde*** `|` ***Numéro*** `, mask:` ***Numéro*** `)`

Constructeur pour la classe IPAddress qui prend deux paramètres :

`ip:` ***Corde***

La chaîne d'adresse IP au format CIDR.

`mask:` ***Numéro***

Le masque de sous-réseau facultatif au format numérique, représentant le nombre de bits « 1 » situés le plus à gauche du masque (facultatif).

Méthodes d'instance

`equals(equals:` ***Adresse IP*** `):` ***Booléen***

Effectue un test d'égalité entre les objets IPAddress, comme illustré dans l' exemple suivant :

```
if (Flow.client.ipaddr.toString() === "10.10.10.10")
{ // perform a task }
```

`mask(mask:` ***Numéro*** `):` ***Adresse IP***

Définit le masque de sous-réseau de l'objet IPAddress comme illustré dans l'exemple suivant :

```
if ((Flow.ipaddr1.mask(24).toString() === "173.194.33.0")||
(Flow.ipaddr2.mask(24).toString() === "173.194.33.0"))
{Flow.setApplication("My L4 App");}
```

Le `mask` Le paramètre spécifie le masque de sous-réseau dans un format numérique, représentant le nombre de bits « 1 » situés le plus à gauche dans le masque (facultatif).

`toJSON():` ***Corde***

Convertit l'objet IPAddress au format JSON.

`toString():` ***Corde***

Convertit l'objet IPAddress en chaîne imprimable.

Propriétés

`hostNames:` ***Tableau de chaînes***

Tableau de noms d'hôtes associés à l'adresse IP.

`isBroadcast:` ***Booléen***

La valeur est `true` si l'adresse IP est une adresse de diffusion.

`isExternal:` ***Booléen***

La valeur est `true` si l'adresse IP est externe à votre réseau.

`isLinkLocal:` ***Booléen***

La valeur est `true` si l'adresse IP est une adresse locale de lien telle que (169.254.0.0/16).

`isMulticast:` ***Booléen***

La valeur est `true` si l'adresse IP est une adresse de multidiffusion.

`isRFC1918:` ***Booléen***

La valeur est `true` si l'adresse IP appartient à l'une des plages d'adresses IP privées de la RFC1918 (10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16). La valeur est toujours `false` pour les adresses IPv6.

`isV4:` ***Booléen***

La valeur est `true` si l'adresse IP est une adresse IPv4.

`isV6:` ***Booléen***

La valeur est `true` si l'adresse IP est une adresse IPv6.

`localityName:` ***Corde*** | ***nul***

Le nom de la localité réseau dans laquelle se trouve l'adresse IP. Si l'adresse IP ne se trouve dans aucune localité réseau, la valeur est nulle.

Network

Le Network La classe vous permet d'ajouter des mesures personnalisées au niveau mondial.

Méthodes

`metricAddCount(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de comptage. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Le nom de la métrique de comptage de niveau supérieur.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailCount(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de comptage grâce auquel vous pouvez effectuer une analyse plus approfondie. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur de l'incrément. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir la propriété suivante :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDataset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique du jeu de données. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Nom de la métrique du jeu de données de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailDataset(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique du jeu de données grâce auquel vous pouvez effectuer une analyse plus approfondie. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Nom de la métrique du nombre de détails.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`freq:` ***Numéro***

Option qui vous permet d'enregistrer simultanément plusieurs occurrences de valeurs particulières dans l'ensemble de données lorsque le nombre d'occurrences est défini par le `val` paramètre. Si aucune valeur n'est spécifiée, la valeur par défaut est 1.

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDistinct(metric_name:` ***Corde*** `, item:` ***Numéro*** | ***Corde*** | ***Adresse IP*** `:void`

Crée une personnalisation niveau supérieur métrique de comptage distincte. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Nom de la métrique de comptage distincte de niveau supérieur.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

MetricAddDetailDistinct (metric_name) : ***Corde*****, clé :** ***Corde*** **|** ***Adresse IP*****, article :** ***Numéro*** **|** ***Corde*** **|** ***Adresse IP*****: nul**

Crée une personnalisation détail métrique de comptage distincte par lequel vous pouvez approfondir. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Nom de la métrique de comptage distincte détaillée.

`key:` ***Corde*** | ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`item:` ***Numéro*** | ***Corde*** | ***Adresse IP***

La valeur à placer dans l'ensemble. La valeur est convertie en chaîne avant d'être placée dans l'ensemble.

`metricAddMax(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique maximale. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Le nom de la métrique maximale de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailMax(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique maximale grâce auquel vous pouvez effectuer une analyse plus approfondie. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Nom de la métrique maximale de détail.

`key:` ***Corde | Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSampleset(metric_name:` ***Corde*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur Sampleset métrique. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Le nom de la métrique de l'ensemble d'échantillons de niveau supérieur.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSampleset(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, val:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail Sampleset métrique par lequel vous pouvez approfondir. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde | Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`val:` ***Numéro***

La valeur observée, telle qu'un temps de traitement. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddSnap(metric_name:` ***Corde*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation niveau supérieur métrique de capture d'écran. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Nom de la métrique de capture instantanée de niveau supérieur.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

`metricAddDetailSnap(metric_name:` ***Corde*** `, key:` ***Corde*** `|` ***Adresse IP*** `, count:` ***Numéro*** `, options:` ***Objet*** `):void`

Crée une personnalisation détail métrique de capture d'écran grâce auquel vous pouvez effectuer une analyse plus approfondie. Valide les données métriques sur le réseau spécifié.

`metric_name:` ***Corde***

Nom de la métrique détaillée de l'ensemble d'échantillons.

`key:` ***Corde*** `|` ***Adresse IP***

Clé spécifiée pour la métrique détaillée. UN `null` la valeur est ignorée silencieusement.

`count:` ***Numéro***

La valeur observée, telle que les connexions actuellement établies. Il doit s'agir d'un entier de 64 bits signé positif différent de zéro. UN `NaN` la valeur est ignorée silencieusement.

`options:` ***Objet***

Un objet facultatif qui peut contenir les propriétés suivantes :

`highPrecision:` ***Booléen***

Un indicateur qui active une granularité en une seconde pour la métrique personnalisée lorsqu'elle est définie sur `true`.

Exemples de déclencheurs

- Exemple : analyse du syslog sur TCP avec une analyse de charge utile universelle
- Exemple : enregistrer des données dans une table de session
- Exemple : suivre les requêtes SOAP

Session

Le Session la classe donne accès à la table de session. Il est conçu pour faciliter la coordination entre plusieurs déclencheurs s'exécutant indépendamment. L'état global de la table de session signifie que toute modification apportée par un déclencheur ou un processus externe devient visible pour tous les autres utilisateurs de la table de session. La table de session étant en mémoire, les modifications ne sont pas enregistrées lorsque vous redémarrez le système ExtraHop ou le processus de capture.

Voici quelques informations importantes à connaître au sujet des tables de session :

- La table de session prend en charge les valeurs JavaScript ordinaires, ce qui vous permet d'ajouter des objets JS à la table.
- Les entrées de la table de session peuvent être supprimées lorsque la table devient trop grande ou lorsque l'expiration configurée est atteinte.
- Parce que la table de session sur un sonde n'est pas partagé avec console, les valeurs de la table de session ne sont pas partagées avec les autres utilisateurs connectés capteurs.
- L'API ExtraHop Open Data Context expose le tableau des sessions via le réseau de gestion, permettant ainsi la coordination avec les processus externes via le cache mémoire protocole.

Évènements

La classe Session ne se limite pas uniquement aux SESSION_EXPIRE événement. Vous pouvez appliquer la classe Session à n'importe quel événement ExtraHop.

SESSION_EXPIRE

S'exécute périodiquement (par incréments d'environ 30 secondes) tant que la table de session est utilisée. Lorsque le SESSION_EXPIRE un événement se déclenche, les clés expirées au cours des 30 secondes précédentes sont disponibles via le Session.expiredKeys propriété.

Le SESSION_EXPIRE l'événement n'est associé à aucun flux particulier, il se déclenche donc SESSION_EXPIRE les événements ne peuvent pas valider les métriques de l'équipement via Device.metricAdd*() méthodes ou Flow.client.device.metricAdd*() méthodes. Pour valider les métriques de l'équipement lors de cet événement, vous devez ajouter Device objets vers le tableau de session par le biais du Device() méthode d'instance.

Note: Vous ne pouvez pas attribuer des déclencheurs qui s'exécutent uniquement lors de cet événement à des appareils ou à des groupes d'équipements spécifiques. Les déclencheurs qui s'exécutent lors de cet événement seront exécutés chaque fois que cet événement se produira.

TIMER_30SEC

Fonctionne exactement toutes les 30 secondes. Cet événement vous permet d'effectuer des traitements périodiques, tels que l'accès régulier aux entrées de table de session ajoutées via le API de contexte de données ouvertes.

Note: Vous pouvez appliquer n'importe quelle classe de déclencheur à l'événement TIMER_30SEC.

Note: Vous ne pouvez pas attribuer des déclencheurs qui s'exécutent uniquement lors de cet événement à des appareils ou à des groupes d'équipements spécifiques. Les déclencheurs qui s'exécutent lors de cet événement seront exécutés chaque fois que cet événement se produira.

Méthodes

add(key: ***Corde***, value*, options: ***Objet***): *

Ajoute la clé spécifiée dans le tableau de session. Si la clé est présente, la valeur correspondante est renvoyée sans modifier l'entrée clé dans le tableau. Si la clé n'est pas présente, une nouvelle entrée est créée pour la clé et la valeur, et la nouvelle valeur est renvoyée.

Vous pouvez configurer une option Des options objet pour la clé spécifiée.

`getOptions(key:` ***Corde*** `):` ***Objet***

Renvoie le Des options objet pour la clé spécifiée. Vous configurez les options lors des appels à `Session.add()`, `Session.modify()`, ou `Session.replace()`.

`increment(key:` ***Corde*** `, count:` ***Numéro*** `):` ***Numéro*** `|` ***nul***

Recherche la clé spécifiée et incrémente la valeur de la clé du nombre spécifié. La valeur par défaut du paramètre de comptage facultatif est 1. Renvoie la nouvelle valeur de la clé si l' appel est réussi. Retours `null` si la recherche échoue. Renvoie une erreur si la valeur de la clé n'est pas un nombre.

`lookup(key:` ***Corde*** `): *`

Recherche la clé spécifiée dans la table de session et renvoie la valeur correspondante. Retours `null` si la clé n'est pas présente.

`modify(key:` ***Corde*** `, value: *, options:` ***Objet*** `): *`

Modifie la valeur de clé spécifiée, si la clé est présente dans la table de session, et renvoie la valeur précédente. Si la clé n'est pas présente, aucune nouvelle entrée n'est créée.

En cas de modification de l'option Des options les objets sont inclus, les options clés sont mises à jour. et les anciennes options sont fusionnées avec les nouvelles. Si le `expire` l'option est modifiée, le délai d'expiration est réinitialisé.

`remove(key:` ***Corde*** `): *`

Supprime l'entrée pour la clé donnée et renvoie la valeur associée.

`replace(key:` ***Corde*** `, value: *, options:` ***Objet*** `): *`

Met à jour l'entrée associée à la clé donnée. Si la clé est présente, mettez à jour la valeur et renvoyez la valeur précédente. Si la clé n'est pas présente, ajoutez l'entrée et renvoyez la valeur précédente (null).

En cas de modification de l'option Des options les objets sont inclus, les options clés sont mises à jour et les anciennes options sont fusionnées avec les nouvelles. Si le `expire` l'option est fournie, le délai d'expiration est réinitialisé.

Des options

`expire:` ***Numéro***

Durée après laquelle l'expulsion a lieu, exprimée en secondes. Si la valeur est `null` ou `undefined`, l'entrée n'est supprimée que lorsque la table de session devient trop grande.

`notify:` ***Booléen***

Indique si la clé est disponible sur `SESSION_EXPIRE` événements. La valeur par défaut est false.

`priority:` ***Corde***

Niveau de priorité qui détermine les entrées à expulser si la table de session devient trop grande. Les valeurs valides sont `PRIORITY_LOW`, `PRIORITY_NORMAL`, et `PRIORITY_HIGH`. La valeur par défaut est `PRIORITY_NORMAL`.

Constantes

`PRIORITY_LOW:` ***Numéro***

Représentation numérique du niveau de priorité le plus bas. La valeur est 0. Les niveaux de priorité déterminent l'ordre dans lequel les entrées sont supprimées de la table de session si celle-ci devient trop grande.

`PRIORITY_NORMAL:` ***Numéro***

Représentation numérique du niveau de priorité par défaut. La valeur est 1. Les niveaux de priorité déterminent l'ordre dans lequel les entrées sont supprimées de la table de session si celle-ci devient trop grande.

`PRIORITY_HIGH:` ***Numéro***

Représentation numérique du niveau de priorité le plus élevé. La valeur est 2. Les niveaux de priorité déterminent l'ordre dans lequel les entrées sont supprimées de la table de session si celle-ci devient trop grande.

Propriétés

`expiredKeys:` ***Array***

Tableau d'objets présentant les propriétés suivantes :

`age:` ***Numéro***

Âge de l'objet expiré, exprimé en millisecondes. L'âge est le temps écoulé entre le moment où l'objet de la table de session a été ajouté ou l' option d'expiration de l'objet a été modifiée, et le `SESSION_EXPIRE` événement. L'âge détermine si la clé a été expulsée ou a expiré.

`name:` ***Corde***

La clé de l'objet expiré.

`value:` ***Numéro*** | ***Corde*** | ***Adresse IP*** | ***Booléen*** | ***Appareil***

La valeur de l'entrée dans le tableau de session.

Les clés expirées incluent les clés qui ont été expulsées parce que la table est devenue trop grande.

Le `expiredKeys` la propriété n'est accessible que sur `SESSION_EXPIRE` événements ; dans le cas contraire, une erreur se produira.

Exemples de déclencheurs

- Exemple : enregistrer des données dans une table de session

`System`

Le `System` la classe vous permet de récupérer des informations sur le sonde ou console sur lequel un déclencheur est en cours d'exécution. Ces informations sont utiles dans les environnements comportant plusieurs capteurs.

Propriétés

`uuid:` ***Corde***

L'identifiant unique universel (UUID) du sonde ou console.

`ipaddr:` ***Adresse IP***

Le IPAddress objet de l' interface de gestion principale (Interface 1) sur la sonde.

`hostname:` ***Corde***

Le nom d'hôte du sonde ou console configuré dans les paramètres d'administration.

`version:` ***Corde***

La version du microprogramme exécutée sur le sonde ou console.

`ThreatIntel`

Le `ThreatIntel` La classe vous permet de voir si des menaces ont été détectées pour les adresses IP, les noms d'hôte ou les URI. (ExtraHop RevealX Premium et Ultra uniquement)

Méthodes

`hasIP(address:` ***Adresse IP*** `):` ***booléen***

La valeur est `true` si les menaces ont été détectées pour l' adresse IP spécifiée. Si aucune information de renseignement n'est disponible sur le système ExtraHop, la valeur est `null`.

`hasDomain(domain:` ***Corde*** `):` ***booléen***

La valeur est `true` si les menaces ont été détectées pour le domaine spécifié. Si aucune information de renseignement n'est disponible sur le système ExtraHop, la valeur est `null`.

`hasURI(uri:` ***Corde*** `):` ***booléen***

La valeur est `true` si les menaces ont été détectées pour l' URI spécifiée. Si aucune information de renseignement n'est disponible sur le système ExtraHop, la valeur est `null`.

Propriétés

`isAvailable:` ***booléen***

La valeur est `true` si des renseignements sur les menaces sont disponibles sur le système ExtraHop.

`Trigger`

Le `Trigger` La classe vous permet d'accéder aux détails d'un déclencheur en cours d'exécution.

Propriétés

`isDebugEnabled:` ***booléen***

La valeur est `true` si le débogage est activé pour le déclencheur. La valeur est déterminée par l'état du **Activer le journal de débogage** case à cocher dans le volet Modifier le déclencheur du système ExtraHop.

`VLAN`

Le `VLAN` la classe représente un VLAN sur le réseau.

Propriétés de l'instance

`id:` ***Numéro***

L'ID numérique d'un VLAN.

Classes de données de protocole et de réseau

Les classes d'API Trigger présentées dans cette section vous permettent d'accéder aux propriétés et d'enregistrer les métriques de protocole, message et activité de flux qui se produit sur le système ExtraHop ExtraHop.

Classe	Descriptif
AAA	Vous permet de stocker des métriques et d'accéder aux propriétés sur AAA_REQUEST ou AAA_RESPONSE événements.
ActiveMQ	Vous permet de stocker des métriques et d'accéder aux propriétés sur ACTIVEMQ_MESSAGE événements.
AJP	La classe AJP vous permet de stocker des métriques et d'accéder aux propriétés sur AJP_REQUEST et AJP_RESPONSE événements.
CDP	La classe CDP vous permet de stocker des métriques et des propriétés d'accès sur CDP_FRAME événements.
CIFS	Vous permet de stocker des métriques et d'accéder aux propriétés sur CIFS_REQUEST et CIFS_RESPONSE événements.
DB	Vous permet de stocker des métriques et d'accéder aux propriétés sur DB_REQUEST et DB_RESPONSE événements.
DHCP	Vous permet de stocker des métriques et d'accéder aux propriétés sur DHCP_REQUEST et DHCP_RESPONSE événements.
DICOM	Vous permet de stocker des métriques et d'accéder aux propriétés sur DICOM_REQUEST et DICOM_RESPONSE événements.
DNS	Vous permet de stocker des métriques et d'accéder aux propriétés sur DNS_REQUEST et DNS_RESPONSE événements.
FIX	Vous permet de stocker des métriques et d'accéder aux propriétés sur FIX_REQUEST et FIX_RESPONSE événements.
FTP	Vous permet de stocker des métriques et d'accéder aux propriétés sur FTP_REQUEST et FTP_RESPONSE événements.
HL7	Vous permet de stocker des métriques et d'accéder aux propriétés sur HL7_REQUEST et HL7_RESPONSE événements.
HTTP	Vous permet de stocker des métriques et d'accéder aux propriétés sur HTTP_REQUEST et HTTP_RESPONSE événements.

Classe	Descriptif
IBMMQ	Vous permet de stocker des métriques et d'accéder aux propriétés sur IBMMQ_REQUEST et IBMMQ_RESPONSE événements.
ICA	Vous permet de stocker des métriques et d'accéder aux propriétés sur ICA_OPEN, ICA_AUTH, ICA_TICK, et ICA_CLOSE événements.
ICMP	Vous permet de stocker des métriques et d'accéder aux propriétés sur ICMP_MESSAGE événements.
Kerberos	Vous permet de stocker des métriques et d'accéder aux propriétés sur KERBEROS_REQUEST et KERBEROS_RESPONSE événements.
LDAP	Vous permet de stocker des métriques et d'accéder aux propriétés sur LDAP_REQUEST et LDAP_RESPONSE événements.
LLDP	Vous permet d'accéder aux propriétés sur LLDP_FRAME événements.
Memcache	Vous permet de stocker des métriques et d'accéder aux propriétés sur MEMCACHE_REQUEST et MEMCACHE_RESPONSE événements.
Modbus	Vous permet de stocker des métriques et d'accéder aux propriétés sur MODBUS_REQUEST et MODBUS_RESPONSE événements.
MongoDB	La classe MongoDB vous permet de stocker des métriques et d'accéder aux propriétés sur MONGODB_REQUEST et MONGODB_RESPONSE événements.
MSMQ	La classe MSMQ vous permet de stocker des métriques et d'accéder aux propriétés sur MSMQ_MESSAGE événement.
NetFlow	Vous permet de stocker des métriques et d'accéder aux propriétés sur NETFLOW_RECORD événements.
NFS	Vous permet de stocker des métriques et d'accéder aux propriétés sur NFS_REQUEST et NFS_RESPONSE événements.
NTLM	Vous permet de stocker des métriques et d'accéder aux propriétés sur NTLM_MESSAGE événements.
POP3	Vous permet de stocker des métriques et d'accéder aux propriétés sur POP3_REQUEST et POP3_RESPONSE événements.
RDP	Vous permet de stocker des métriques et d'accéder aux propriétés sur RDP_OPEN, RDP_CLOSE, et RDP_TICK événements.
Redis	Vous permet de stocker des métriques et d'accéder aux propriétés sur REDIS_REQUEST et REDIS_RESPONSE événements.

Classe	Descriptif
RPC	Vous permet de stocker des métriques et d'accéder aux propriétés sur RPC_REQUEST et RPC_RESPONSE événements.
RTCP	Vous permet de stocker des métriques et d'accéder aux propriétés sur RTCP_MESSAGE événements.
RTP	Vous permet de stocker des métriques et d'accéder aux propriétés sur RTP_OPEN, RTP_CLOSE, et RTP_TICK événements.
SCCP	Vous permet de stocker des métriques et d'accéder aux propriétés sur SCCP_MESSAGE événements.
SDP	Vous permet d'accéder aux propriétés sur SIP_REQUEST et SIP_RESPONSE événements.
SFlow	Vous permet de stocker des métriques et d'accéder aux propriétés sur SFLOW_RECORD événements.
SIP	Vous permet de stocker des métriques et d'accéder aux propriétés sur SIP_REQUEST et SIP_RESPONSE événements.
SMPP	Vous permet de stocker des métriques et d'accéder aux propriétés sur SMPP_REQUEST et SMPP_RESPONSE événements.
SMTP	Vous permet de stocker des métriques et d'accéder aux propriétés sur SMTP_REQUEST et SMTP_RESPONSE événements.
SSH	Vous permet de stocker des métriques et d'accéder aux propriétés sur SSH_CLOSE, SSH_OPEN et SSH_TICK événements.
SSL	Vous permet de stocker des métriques et d'accéder aux propriétés sur SSL_OPEN, SSL_CLOSE, SSL_ALERT, SSL_RECORD, SSL_HEARTBEAT, et SSL_RENEGOTIATE événements.
TCP	Vous permet d'accéder aux propriétés et de récupérer des métriques à partir d'événements TCP, etc. FLOW_TICK et FLOW_TURN événements.
Telnet	Vous permet de stocker des métriques et d'accéder aux propriétés sur TELNET_MESSAGE événements.
Turn	Vous permet de stocker des métriques et d'accéder aux propriétés sur FLOW_TURN événements.
UDP	Vous permet d'accéder aux propriétés et de récupérer des métriques à partir d'événements UDP, etc. FLOW_TICK et FLOW_TURN événements.
WebSocket	Vous permet d'accéder aux propriétés sur WEBSOCKET_OPEN, WEBSOCKET_CLOSE, et WEBSOCKET_MESSAGE événements.

AAA

Le AAA La classe (Authentication, autorisation et comptabilité) vous permet de stocker des métriques et d'accéder aux propriétés sur AAA_REQUEST ou AAA_RESPONSE événements.

Évènements

AAA_REQUEST

S'exécute lorsque le système ExtraHop a fini de traiter une demande AAA.

AAA_RESPONSE

Fonctionne sur chaque réponse AAA traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un AAA_REQUEST ou AAA_RESPONSE événement.

L'événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si le `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`authenticator:` ***Corde***

La valeur du champ d'authentificateur (RADIUS uniquement).

`avps:` ***Array***

Un tableau d'objets AVP présentant les propriétés suivantes :

`avpLength:` ***Numéro***

La taille de l'AVP, exprimée en octets. Cette valeur inclut les données d'en-tête AVP, ainsi que la valeur.

`id:` ***Numéro***

L'ID numérique de l'attribut représenté sous forme de nombre entier.

`isGrouped:` ***Booléen***

La valeur est `true` s'il s'agit d'un AVP groupé (diamètre uniquement).

`name:` ***Corde***

Le nom de l'AVP donné.

`vendor:` ***Corde***

Le nom du fournisseur pour les AVP du fournisseur (Diameter uniquement).

`value:` ***Corde*** | ***Array*** | ***Numéro***

Pour les AVP uniques, une chaîne ou une valeur numérique. Pour les AVP groupés (diamètre uniquement), un tableau d'objets.

`isDiameter:` ***Booléen***

La valeur est `true` si la demande ou la réponse est Diameter.

`isError:` ***Booléen***

La valeur est `true` si la réponse est une erreur. Pour récupérer le détail de l'erreur dans Diameter, vérifiez `AAA.statusCode`. Pour récupérer les détails de l'erreur dans RADIUS, vérifiez l'AVP avec le code 18 (message de réponse).

Accès uniquement sur AAA_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`isRadius:` ***Booléen***

La valeur est `true` si la demande ou la réponse est RADIUS.

`isRspAborted:` ***Booléen***

La valeur est `true` si le AAA_RESPONSE l'événement est annulé.

Accès uniquement sur AAA_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`method:` ***Numéro***

Méthode correspondant au code de commande dans RADIUS ou Diameter.

Le tableau suivant contient les codes de commande Diameter valides :

Nom de la commande	Abbé.	Code
AA-Request	AAR	265
AA-Answer	AAA	265
Diameter-EAP-Request	DER	268
Diameter-EAP-Answer	DEA	268
Abort-Session-Request	ASR	274
Abort-Session-Answer	ASA	274
Accounting-Request	ACR	271
Credit-Control-Request	CCR	272
Credit-Control-Answer	CCA	272
Capabilities-Exchange-Request	CER	257
Capabilities-Exchange-Answer	CEA	257
Device-Watchdog-Request	DWR	280
Device-Watchdog-Answer	DWA	280
Disconnect-Peer-Request	DPR	282
Disconnect-Peer-Answer	DPA	282
Re-Auth-Answer	RAA	258
Re-Auth-Request	RAR	258
Session-Termination-Request	STR	275
Session-Termination-Answer	STA	275
User-Authorization-Request	UAR	300
User-Authorization-Answer	UAA	300
Server-Assignment-Request	SAR	301
Server-Assignment-Answer	SAA	301
Location-Info-Request	LIR	302
Location-Info-Answer	LIA	302
Multimedia-Auth-Request	MAR	303
Multimedia-Auth-Answer	MAA	303

Nom de la commande	Abbé.	Code
Registration-Termination-Request	RTR	304
Registration-Termination-Answer	RTA	304
Push-Profile-Request	PPR	305
Push-Profile-Answer	PPA	305
User-Data-Request	UDR	306
User-Data-Answer	UDA	306
Profile-Update-Request	PUR	307
Profile-Update-Answer	PUA	307
Subscribe-Notifications-Request	SNR	308
Subscribe-Notifications-Answer	SNA	308
Push-Notification-Request	PNR	309
Push-Notification-Answer	PNA	309
Bootstrapping-Info-Request	BIR	310
Bootstrapping-Info-Answer	BIA	310
Message-Process-Request	MPR	311
Message-Process-Answer	MPA	311
Update-Location-Request	ULR	316
Update-Location-Answer	ULA	316
Authentication-Information-Request	AIR	318
Authentication-Information-Answer	AIA	318
Notify-Request	NR	323
Notify-Answer	NA	323

Le tableau suivant contient des codes de commande RADIUS valides :

Nom de la commande	Code
Access-Request	1
Access-Accept	2
Access-Reject	3
Accounting-Request	4
Accounting-Response	5
Access-Challenge	11
Status-Server (experimental)	12
Status-Client (experimental)	13
Reserved	255

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. La valeur est `NaN` si le chronométrage n'est pas valide.

Accès uniquement sur AAA_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `AAA.commitRecord()` sur l'un ou l'autre AAA_REQUEST ou AAA_RESPONSE événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

AAA_REQUEST	AAA_RESPONSE
authenticator	authenticator
clientIsExternal	clientIsExternal
clientZeroWnd	clientZeroWnd
method	isError
receiverIsExternal	isRspAborted
reqBytes	method
reqL2Bytes	processingTime
reqPkts	receiverIsExternal
reqRTO	roundTripTime
senderIsExternal	rspBytes
serverIsExternal	rspL2Bytes
serverZeroWnd	rspPkts
txId	rspRTO
	statusCode
	senderIsExternal
	serverIsExternal
	serverZeroWnd
	txId

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

Accès uniquement sur AAA_REQUEST événements ; dans le cas contraire, une erreur se produira.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, comme les en-têtes.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

Accès uniquement sur `AAA_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`statusCode:` ***Corde***

Une représentation sous forme de chaîne de l'identifiant AVP 268 (code de résultat).

Accès uniquement sur `AAA_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`txId:` ***Numéro***

Une valeur qui correspond à l'identifiant saut par saut dans Diameter et à l'identifiant msg-id dans RADIUS.

`ActiveMQ`

Le ActiveMQ la classe vous permet de stocker des métriques et d'accéder à des propriétés sur `ACTIVEMQ_MESSAGE` événements. ActiveMQ est une implémentation du service de messagerie Java (JMS).

Évènements

`ACTIVEMQ_MESSAGE`

S'exécute sur tous les messages JMS traités par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `ACTIVEMQ_MESSAGE` événement.

Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`correlationId:` ***Corde***

Le champ JMScorrelationID du message.

`exceptionResponse:` ***Objet | Null***

Le champ JMSException du message. Si la commande du message n'est pas `ExceptionResponse`, la valeur est nulle. L'objet contient les champs suivants :

`message:` ***Corde***

Le message de réponse à l'exception.

`class:` ***Corde***

La sous-classe de la JMSException.

`expiration:` ***Numéro***

Le champ JMSExpiration du message.

`msg:` ***Tampon***

Le corps du message. Pour les messages au format TEXT_MESSAGE, le corps du message est renvoyé sous la forme d'une chaîne UTF-8. Pour tous les autres formats de message, cela renvoie les octets bruts.

`msgFormat:` ***Corde***

Format du message. Les valeurs possibles sont les suivantes :

- `BYTES_MESSAGE`
- `MAP_MESSAGE`
- `MESSAGE`
- `OBJECT_MESSAGE`
- `STREAM_MESSAGE`
- `TEXT_MESSAGE`
- `BLOG_MESSAGE`

`msgId:` ***Corde***

Le champ JMSMessageID du message.

`persistent:` ***Booléen***

La valeur est `true` si le JMSDeliveryMode est PERSISTANT.

`priority:` ***Numéro***

Le champ JMSPriority du message.

- 0 est la priorité la plus basse.
- 9 est la priorité la plus élevée.
- 0 à 4 sont des gradations de priorité normale.
- 5 à 9 sont des gradations de priorité accélérée.

`properties:` ***Objet***

Zéro ou plusieurs propriétés associées au message. Les clés sont des chaînes arbitraires et les valeurs peuvent être des booléens, des nombres ou des chaînes.

`queue:` ***Corde***

Le champ JMSDestination du message.

`receiverBytes:` ***Numéro***

Le nombre d'octets au niveau de l'application provenant du récepteur.

`receiverIsBroker:` ***Booléen***

La valeur est `true` si le destinataire du message au niveau du flux est un courtier.

`receiverL2Bytes:` ***Numéro***

Le nombre de L2 octets provenant du récepteur.

`receiverPkts:` ***Numéro***

Le nombre de paquets provenant du récepteur.

`receiverRTO:` ***Numéro***

Le nombre de RTO émis par le récepteur.

`receiverZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par le récepteur.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `ActiveMQ.commitRecord()` sur un `ACTIVEMQ_MESSAGE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `correlationId`
- `expiration`
- `msgFormat`
- `msgId`
- `persistent`
- `priority`
- `queue`
- `receiverBytes`
- `receiverIsBroker`
- `receiverIsExternal`
- `receiverL2Bytes`
- `receiverPkts`
- `receiverRTO`
- `receiverZeroWnd`
- `redeliveryCount`
- `replyTo`
- `roundTripTime`
- `senderBytes`
- `senderIsBroker`
- `senderIsExternal`
- `senderL2Bytes`
- `senderPkts`
- `senderRTO`
- `senderZeroWnd`
- `serverIsExternal`
- `timeStamp`
- `totalMsgLength`

`redeliveryCount:` ***Numéro***

Le nombre de relivraisons.

`replyTo:` ***Corde***

Le champ JMSReplyTo du message, converti en chaîne.

`roundTripTime:` ***Numéro***

Le temps médian aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`senderBytes:` ***Numéro***

Le nombre d'octets au niveau de l'application provenant de l'expéditeur.

`senderIsBroker:` ***Booléen***

La valeur est `true` si l'expéditeur du message au niveau du flux est un courtier.

`senderL2Bytes:` ***Numéro***

Le nombre de L2 octets provenant de l'expéditeur.

`senderPkts:` ***Numéro***

Le nombre de paquets provenant de l'expéditeur.

`senderRTO:` ***Numéro***

Le nombre de RTO émis par l'expéditeur.

`senderZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par l'expéditeur.

`timestamp:` ***Numéro***

Heure à laquelle le message a été transmis à un fournisseur pour être envoyé, exprimée en GMT. Il s'agit du champ JMSTimestamp du message.

`totalMsgLength:` ***Numéro***

Longueur du message, exprimée en octets.

AJP

Le protocole AJP (Apache JServ Protocol) transmet par proxy les requêtes entrantes d'un serveur Web à un serveur d'applications et est souvent appliqué à des environnements à charge équilibrée dans lesquels un ou plusieurs serveurs Web frontaux transmettent des demandes à un ou plusieurs serveurs d'applications. Le AJP la classe vous permet de stocker des métriques et d'accéder à des propriétés sur AJP_REQUEST et AJP_RESPONSE événements.

Évènements

AJP_REQUEST

S'exécute une fois que le serveur Web a envoyé un message de demande de transfert AJP à un conteneur de servlets, puis a transféré le corps de la requête suivant.

AJP_RESPONSE

S'exécute lorsqu'un conteneur de servlet envoie un message de réponse de fin AJP pour signaler que le conteneur de servlet a fini de traiter une demande de transfert AJP et a renvoyé les informations demandées.

Méthodes

`commitRecord():` ***Vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un AJP_RESPONSE événement. Enregistrez les validations le AJP_REQUEST les événements ne sont pas pris en charge.

Pour afficher les propriétés par défaut validées pour l' objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

`findHeaders(name:` ***Corde*** `):` ***Array***

Accède aux valeurs d'en-tête AJP et renvoie un tableau d'objets d'en-tête (avec des propriétés de nom et de valeur) dont les noms correspondent au préfixe de la chaîne spécifiée. Accède aux en-têtes de demande sur AJP_REQUEST événements et en-têtes de réponse sur AJP_RESPONSE demandes.

Propriétés

`attributes:` ***Objet***

Tableau d'attributs AJP facultatifs envoyés avec la demande, tels que remote_user, auth_type, query_string, jvm_route, ssl_cert, ssl_cipher et ssl_session.

Accès uniquement sur `AJP_REQUEST` événements ; sinon, une erreur se produira.

`fwdReqClientAddr:` ***Adresse IP***

Le IPAddress du client HTTP qui a envoyé la demande initiale au serveur. La valeur est `null` si les informations disponibles ne peuvent pas être analysées vers une adresse IP.

`fwdReqHost:` ***Corde***

L'hôte HTTP spécifié par le client HTTP qui a envoyé la demande initiale au serveur.

`fwdReqIsEncrypted:` ***Booléen***

La valeur est `true` si le chiffrement TLS a été appliqué par le client HTTP qui a envoyé la demande initiale au serveur.

`fwdReqServerName:` ***Corde***

Le nom du serveur auquel le client HTTP a envoyé la demande initiale.

Port du serveur FWDREQ : ***Numéro***

Port TCP du serveur auquel le client HTTP a fait la demande initiale.

`headers:` ***Objet***

Lors de l'accès sur `AJP_REQUEST` events, un tableau de noms d'en-têtes et de valeurs envoyés avec la demande.

Lors de l'accès sur `AJP_RESPONSE` events, un tableau d'en-têtes transmis dans le message AJP Send Headers par le serveur au navigateur de l'utilisateur final .

`method:` ***Corde***

Méthode HTTP de la requête, telle que POST ou GET, envoyée du serveur au conteneur de servlets.

`processingTime:` ***Numéro***

Temps entre le dernier octet de la demande reçue et le premier octet de la charge utile de réponse envoyée, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées et abandonnées ou si le timing n' est pas valide.

Accès uniquement sur `AJP_RESPONSE` événements ; sinon, une erreur se produira.

`protocol:` ***Corde***

Le protocole de la requête envoyée par le serveur au conteneur de servlets. Non défini pour les autres types de messages.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `AJP.commitRecord()` sur un `AJP_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `fwdReqClientAddr`
- `fwdReqHost`
- `fwdReqIsEncrypted`
- `fwdReqServerName`
- `fwdReqServerPort`
- `method`
- `processingTime`
- `protocol`
- `receiverIsExternal`
- `reqSize`

- rspSize
- statusCode
- senderIsExternal
- serverIsExternal
- uri

Accès uniquement sur AJP_RESPONSE événements ; sinon, une erreur se produira.

reqBytes: ***Numéro***

Le nombre de L4 octets de requête, à l'exclusion des en-têtes L4.

Accès uniquement sur AJP_RESPONSE événements ; sinon, une erreur se produira.

reqL2Bytes: ***Numéro***

Le nombre de L2 octets de requête, y compris les en-têtes L2.

reqPkts: ***Numéro***

Le nombre de paquets de requêtes.

reqRTO: ***Numéro***

Le numéro de demande délais de retransmission (RTO).

reqSize: ***Numéro***

Nombre d'octets de requête L7, à l'exclusion des en-têtes AJP.

rspBytes: ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur AJP_RESPONSE événements ; sinon, une erreur se produira.

rspL2Bytes: ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

Accès uniquement sur AJP_RESPONSE événements ; sinon, une erreur se produira.

rspPkts: ***Numéro***

Le nombre de paquets de réponse.

Accès uniquement sur AJP_RESPONSE événements ; sinon, une erreur se produira.

rspRTO: ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

Accès uniquement sur AJP_RESPONSE événements ; sinon, une erreur se produira.

rspSize: ***Numéro***

Nombre d'octets de réponse L7, à l'exclusion des en-têtes AJP.

Accès uniquement sur AJP_RESPONSE événements ; sinon, une erreur se produira.

statusCode: ***Numéro***

Le code d'état HTTP renvoyé par le conteneur de servlets pour les réponses aux messages AJP Forward Request.

Accès uniquement sur AJP_RESPONSE événements ; sinon, une erreur se produira.

uri: ***Corde***

L'URI de la demande envoyée par le serveur au conteneur de servlets. Non défini pour les types de messages non AJP.

CDP

Le Cisco Discovery Protocol (CDP) est un protocole propriétaire qui permet aux appareils Cisco connectés de s'envoyer des informations entre eux. Le `CDP` la classe vous permet d' accéder aux propriétés sur `CDP_FRAME` événements.

Évènements

`CDP_FRAME`

Fonctionne sur chaque trame CDP traitée par l'équipement.

Propriétés

`destination:` ***Corde***

Adresse MAC de destination. La destination la plus courante est `01:00:0c:cc:cc:cc`, indiquant une adresse de multidiffusion.

`checksum:` ***Numéro***

La somme de contrôle du CDP.

`source:` ***Appareil***

L'équipement qui envoie la trame CDP.

`ttl:` ***Numéro***

Le temps de vie, exprimé en secondes. Il s'agit de la durée pendant laquelle les informations contenues dans ce cadre sont valides, à compter de leur réception.

`tlvs:` ***Tableau d'objets***

Un tableau contenant chaque champ de type, de longueur et de valeur (TLV). Un champ TLV contient des informations telles que l'identifiant, l'adresse et la plate-forme de l'équipement. Chaque champ est un objet doté des propriétés suivantes :

`type:` ***Numéro***

Le type de TLV.

`value:` ***Tampon***

La valeur du TLV.

`version:` ***Numéro***

Version du protocole CDP.

CIFS

Le `CIFS` la classe vous permet de stocker des métriques et d'accéder à des propriétés sur `CIFS_REQUEST` et `CIFS_RESPONSE` événements.

Évènements

`CIFS_REQUEST`

S'exécute sur chaque demande SMB traitée par l'équipement.

`CIFS_RESPONSE`

S'exécute sur chaque réponse SMB traitée par l'équipement.

Note: Le `CIFS_RESPONSE` l'événement se déroule après chaque `CIFS_REQUEST` événement, même si la réponse correspondante n'est jamais observée par le système ExtraHop .

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur `CIFS_RESPONSE` événement. Enregistrez les validations le `CIFS_REQUEST` les événements ne sont pas pris en charge.

Pour afficher les propriétés par défaut validées pour l' objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

Important: Le temps d'accès est le temps nécessaire à un serveur SMB pour recevoir un bloc demandé. Il n'y a pas de temps d'accès pour les opérations qui n'accèdent pas aux données de bloc réelles d'un fichier. Le temps de traitement est le temps nécessaire à un serveur SMB pour répondre à l' opération demandée par le client, telle qu'une demande de récupération de métadonnées.

Il n'existe aucun temps d'accès pour les commandes SMB2_CREATE, qui créent un fichier référencé dans la réponse par une commande SMB2_FILEID. Les blocs de fichiers référencés sont ensuite lus ou écrits sur l'équipement de stockage NAS. Ces opérations de lecture et d'écriture de fichiers sont calculées en tant que temps d'accès.

`accessTime:` ***Numéro***

Temps nécessaire au serveur pour accéder à un fichier sur disque, exprimé en millisecondes. Pour SMB, il s'agit du temps écoulé entre la première commande READ d'un flux SMB et le premier octet de la charge utile de réponse. La valeur est `NaN` si la mesure ou le chronométrage ne sont pas valides.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`dialect:` ***Corde***

Le dialecte du SMB négocié entre le client et le serveur.

`encryptedBytes:` ***Numéro***

Le nombre d'octets chiffrés contenus dans la demande ou la réponse.

`encryptionProtocol:` ***Corde***

Le protocole avec lequel la transaction est cryptée.

`error:` ***Corde***

Message d'erreur détaillé enregistré par le système ExtraHop.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`filename:` ***Corde***

Le nom du fichier en cours de transfert.

`isCommandCreate:` ***Booléen***

La valeur est `true` si le message contient une commande de création de fichier SMB.

`isCommandClose:` ***Booléen***

La valeur est `true` si le message contient une commande SMB CLOSE.

`isCommandDelete:` ***Booléen***

La valeur est `true` si le message contient une commande SMB DELETE.

`isCommandFileInfo:` ***Booléen***

La valeur est `true` si le message contient une commande SMB file info.

`isCommandLock:` ***Booléen***

La valeur est `true` si le message contient une commande de verrouillage SMB.

`isCommandRead:` ***Booléen***

La valeur est `true` si le message contient une commande SMB READ.

`isCommandRename:` ***Booléen***

La valeur est `true` si le message contient une commande SMB RENAME.

`isCommandWrite:` ***Booléen***

La valeur est `true` si le message contient une commande SMB WRITE.

`isDecrypted:` ***Booléen***

La valeur est vraie si le système ExtraHop a déchiffré et analysé la transaction de manière sécurisée. L'analyse du trafic déchiffré peut révéler des menaces avancées qui se cachent dans le trafic chiffré.

`isEncrypted:` ***Booléen***

La valeur est vraie si la transaction est cryptée.

`isRspAborted:` ***Booléen***

La valeur est vraie si la connexion est fermée avant la fin de la réponse SMB .

Accès uniquement sur `CIFS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`isRspSigned:` ***Booléen***

La valeur est vraie si la réponse est signée par le serveur SMB.

`method:` ***Corde***

La méthode SMB. Correspond aux méthodes répertoriées sous la métrique SMB dans le système ExtraHop.

`msgID:` ***Numéro***

L'identifiant de transaction SMB.

`payload:` ***Tampon***

Le Tampon objet contenant les octets de charge utile à partir de la commande READ ou WRITE du message SMB.

La mémoire tampon contient *N* premiers octets de la charge utile, où *N* est le nombre d'octets de charge utile spécifié par Octets de charge utile L7 vers la mémoire tampon option lorsque le déclencheur a été configuré via l'interface utilisateur Web ExtraHop. Le nombre d'octets par défaut est de 2 048. Pour plus d'informations, voir Options de déclencheur avancées.

Note: La mémoire tampon ne peut pas contenir plus de 4 Ko, même si Octets de charge utile L7 vers la mémoire tampon l'option est définie sur une valeur plus élevée.

Pour des volumes plus importants d'octets de charge utile, la charge utile peut être répartie sur une série de commandes READ ou WRITE afin qu'aucun événement déclencheur ne contienne l'intégralité de la charge utile demandée. Vous pouvez réassembler la charge utile en une seule mémoire tampon consolidée via `Flow.store` et `payloadOffset` propriétés.

`payloadMediaType:` ***Corde | Null***

Type de support contenu dans la charge utile. La valeur est nulle s'il n'y a aucune charge utile ou si le type de support est inconnu.

`payloadOffset:` ***Numéro***

Le décalage du fichier, exprimé en octets, dans la `resource` propriété. La propriété de charge utile est obtenue à partir de `resource` propriété au décalage.

`payloadSHA256:` ***Corde | Null***

Représentation hexadécimale du hachage SHA-256 de la charge utile. La chaîne ne contient aucun délimiteur, comme illustré dans l'exemple suivant :

```
468c6c84db844821c9ccb0983c78d1cc05327119b894b5ca1c6a1318784d3675
```

S' il n'y a pas de charge utile, la valeur est nulle.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées et abandonnées ou si le timing n' est pas valide.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `CIFS.commitRecord` sur un `CIFS_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `accessTime`
- `clientIsExternal`
- `clientZeroWnd`
- `error`
- `isCommandCreate`
- `isCommandDelete`
- `isCommandFileInfo`
- `isCommandLock`
- `isCommandRead`
- `isCommandRename`
- `isCommandWrite`
- `isHighEntropy`
- `method`
- `processingTime`
- `receiverIsExternal`
- `reqPayloadMediaType`
- `reqPayloadSHA256`
- `reqSize`
- `reqXfer`
- `resource`
- `rspBytes`
- `rspPayloadMediaType`
- `rspPayloadSHA256`
- `rspXfer`
- `senderIsExternal`
- `serverIsExternal`
- `serverZeroWnd`
- `share`
- `statusCode`
- `user`
- `warning`

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`reqBytes:` ***Numéro***

Le nombre de L4 octets de requête, à l'exclusion des en-têtes L4.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de requête, y compris les en-têtes L2.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`reqPkts:` ***Numéro***

Le nombre de paquets de requêtes.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`reqSize:` ***Numéro***

Nombre d'octets de requête L7, à l'exclusion des en-têtes SMB.

`reqTransferTime:` ***Numéro***

Le temps de transfert de la demande, exprimé en millisecondes. Si la demande est contenue dans un seul paquet, le temps de transfert est nul. Si la demande s'étend sur plusieurs paquets, la valeur est le laps de temps entre la détection du premier paquet de requête SMB et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une demande SMB importante ou un retard du réseau. La valeur est `NaN` s'il n'y a pas de mesure valide ou si le chronométrage n'est pas valide.

Accès uniquement sur `CIFS_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`reqVersion:` ***Corde***

Version de SMB exécutée sur la demande.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`resource:` ***Corde***

Le partage, le chemin et le nom de fichier, concaténés ensemble.

`roundTripTime:` ***Numéro***

Le temps d'aller-retour médian (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`rspSize:` ***Numéro***

Nombre d'octets de réponse L7, à l'exclusion des en-têtes SMB.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`rspTransferTime:` ***Numéro***

Le temps de transfert de la réponse, exprimé en millisecondes. Si la réponse est contenue dans un seul paquet, le temps de transfert est nul. Si la réponse couvre plusieurs paquets, la valeur est la

durée entre la détection du premier paquet de réponse SMB et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une réponse SMB importante ou un retard du réseau. La valeur est `NaN` s'il n'y a pas de mesure valide ou si le chronométrage n'est pas valide.

Accès uniquement sur `CIFS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspVersion:` ***Corde***

Version de SMB exécutée sur la réponse.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`share:` ***Corde***

Le nom du partage auquel l'utilisateur est connecté.

`statusCode:` ***Numéro***

Le code dquo numérique de la réponse (SMB1 et SMB2 uniquement).

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

`user:` ***Corde***

Le nom d'utilisateur, s'il est disponible. Dans certains cas, par exemple lorsque l'événement de connexion n'était pas visible ou que l'accès était anonyme, le nom d'utilisateur n'est pas disponible.

`warning:` ***Corde***

Message d'avertissement détaillé enregistré par le système ExtraHop.

Accès uniquement sur `CIFS_RESPONSE` événements ; sinon, une erreur se produira.

Exemples de déclencheurs

- Exemple : Surveiller les actions des PME sur les appareils

`DB`

Le `DB`, ou base de données, la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `DB_REQUEST` et `DB_RESPONSE` événements.

Évènements

`DB_REQUEST`

S'exécute sur chaque demande de base de données traitée par l'équipement.

`DB_RESPONSE`

S'exécute sur chaque réponse de base de données traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `DB_RESPONSE` événement. Enregistrer les validations sur `DB_REQUEST` les événements ne sont pas pris en charge.

Pour consulter les propriétés par défaut attribuées à l' objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`appName:` ***Corde***

Le client nom de l'application, qui est extrait uniquement pour les connexions MS SQL.

`correlationId:` ***Numéro***

ID de corrélation pour les applications DB2. La valeur est `null` pour les applications autres que DB2.

`database:` ***Corde***

L'instance de base de données. Dans certains cas, par exemple lorsque les événements de connexion sont chiffrés, le nom de la base de données n'est pas disponible.

`encryptionProtocol:` ***Corde***

Le protocole avec lequel la transaction est cryptée.

`error:` ***Corde***

Les messages d'erreur détaillés enregistrés par le système ExtraHop sous forme de chaîne. S'il y a plusieurs erreurs dans une réponse, elles sont concaténées en une seule chaîne.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`errors:` ***Tableau de chaînes***

Les messages d'erreur détaillés enregistrés par le système ExtraHop sous forme de tableau. S'il n' y a qu'une seule erreur dans la réponse, l'erreur est renvoyée sous forme de tableau contenant une chaîne.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`isDecrypted:` ***Booléen***

La valeur est vraie si le système ExtraHop a déchiffré et analysé la transaction en toute sécurité. L'analyse du trafic déchiffré peut révéler les menaces avancées qui se cachent dans le trafic chiffré.

`isEncrypted:` ***Booléen***

La valeur est vraie si la transaction est cryptée.

`isReqAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la demande de base de données ne soit terminée.

`isRspAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la réponse de base de données ne soit terminée.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`method:` ***Corde***

Méthode de base de données en corrélation avec les méthodes répertoriées sous la métrique de base de données dans le système ExtraHop.

`params:` ***Array***

Un tableau d'appels de procédure distants (RPC) paramètres uniquement disponibles pour les bases de données Microsoft SQL, PostgreSQL et DB2.

Le tableau contient chacun des paramètres suivants :

`name:` ***Corde***

Nom facultatif du paramètre RPC fourni.

`value:` ***Corde | Numéro***

Un champ de texte, de nombre entier ou d'heure et de date. Si la valeur n'est pas un champ de texte, de nombre entier ou d' heure et de date, elle est convertie au format HEX/ASCII.

La valeur du `params` la propriété est la même lorsqu'on y accède sur l' un ou l'autre `DB_REQUEST` ou le `DB_RESPONSE` événement.

`procedure:` ***Corde***

Nom de la procédure stockée. Correspond aux procédures répertoriées sous les méthodes de base de données dans le système ExtraHop.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes (équivalent à `rspTimeToFirstByte - reqTimeToLastByte`). La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `DB.commitRecord` sur un `DB_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- Nom de l'application
- Le client est externe
- Client Zerownd
- ID de corrélation
- base de données
- erreur
- est réavorté
- ISRS avorté
- méthode
- procédure
- Le récepteur est externe
- Taille de la requête
- ReqTimeToLastByte
- Taille RSP
- Temps RSP jusqu'au premier octet
- RSPTimeToLastByte
- Délai de traitement
- L'expéditeur est externe
- Le serveur est externe
- Serveur Zerownd
- déclaration
- table
- utilisateur

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`reqSize:` ***Numéro***

Nombre d'octets de requête L7, à l'exclusion des en-têtes de protocole de base de données.

`reqTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la demande, exprimé en millisecondes. Retours `NaN` en cas de demandes mal formées ou abandonnées ou si le délai n'est pas valide.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris les surcharges liées au protocole, telles que les en-têtes.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`rspSize:` ***Numéro***

Nombre d'octets de réponse L7, à l'exclusion des en-têtes de protocole de base de données.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`rspTimeToFirstByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le premier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`rspTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `DB_RESPONSE` événements ; sinon, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`serverVersion:` ***Corde***

Version du serveur MS SQL.

`statement:` ***Corde***

L'instruction SQL complète, qui n'est peut-être pas disponible pour toutes les méthodes de base de données.

`table:` ***Corde***

Nom de la table de base de données spécifiée dans l'instruction en cours. Les bases de données suivantes sont prises en charge :

- `Sybase`
- `Sybase IQ`
- `MySQL`
- `PostgreSQL`
- `IBM Informix`
- `MS SQL TDS`
- `Oracle TNS`
- `DB2`

Renvoie un champ vide si la demande ne contient aucun nom de table.

`user:` ***Corde***

Le nom d'utilisateur, s'il est disponible. Dans certains cas, par exemple lorsque les événements de connexion sont chiffrés, le nom d'utilisateur n'est pas disponible.

Exemples de déclencheurs

- Exemple : collecte des mesures de réponse sur les requêtes de base de données
- Exemple : création d'un conteneur d'applications

`DHCP`

Le DHCP la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `DHCP_REQUEST` et `DHCP_RESPONSE` événements.

Évènements

`DHCP_REQUEST`

S'exécute sur chaque demande DHCP traitée par l'équipement.

`DHCP_RESPONSE`

S'exécute sur chaque réponse DHCP traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `DHCP_REQUEST` ou `DHCP_RESPONSE` événement.

L'événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

`getOption(optionCode:` ***Numéro*** `):` ***Objet***

Accepte un entier du code d'option DHCP en entrée et renvoie un objet contenant les champs suivants :

`code:` ***Numéro***

Le code d'option DHCP.

`name:` ***Corde***

Nom de l'option DHCP.

`payload:` ***Numéro*** | ***Corde***

Le type de charge utile renvoyé sera quel que soit le type de cette option spécifique, comme une adresse IP, un tableau d'adresses IP ou un objet tampon.

Retours `null` si le code d'option spécifié n'est pas présent dans le message.

Propriétés

`chaddr:` ***Corde***

Adresse matérielle du client DHCP.

`clientReqDelay:` ***Numéro***

Le temps écoulé avant le client tente d'acquérir ou de renouveler un bail DHCP, exprimé en secondes.

Accès uniquement sur `DHCP_REQUEST` événements ; sinon, une erreur se produira.

`error:` ***Corde***

Le message d'erreur associé au code d'option 56. La valeur est `null` s'il n'y a pas d'erreur.

Accès uniquement sur `DHCP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`gwAddr:` ***Adresse IP***

L'adresse IP par laquelle les routeurs transmettent les messages de demande et de réponse.

`htype:` ***Numéro***

Le code du type de matériel.

`msgType:` ***Corde***

Le type de message DHCP. Les types de messages pris en charge sont les suivants :

- `DHCPDISCOVER`
- `DHCPOFFER`
- `DHCPREQUEST`
- `DHCPDECLINE`
- `DHCPACK`
- `DHCPNAK`
- `DHCPRELEASE`
- `DHCPINFORM`
- `DHCPFORCERENEW`
- `DHCPLEASEQUERY`
- `DHCPLEASEUNASSIGNED`
- `DHCPLEASEUNKNOWN`
- `DHCPLEASEACTIVE`
- `DHCPBULKLEASEQUERY`
- `DHCPLEASEQUERYDONE`

`offeredAddr:` ***Adresse IP***

L'adresse IP que le serveur DHCP propose ou attribue au client.

Accès uniquement sur `DHCP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`options:` ***Tableau d'objets***

Un tableau d'objets, chaque objet contenant les champs suivants :

`code:` ***Numéro***

Le code d'option DHCP.

`name:` ***Corde***

Nom de l'option DHCP.

`payload:` ***Numéro | Corde***

Le type de charge utile renvoyé sera quel que soit le type de cette option spécifique, comme une adresse IP, un tableau d'adresses IP ou un objet tampon. Les adresses IP seront analysées dans un tableau, mais si le nombre d'octets n'est pas divisible par 4, elles seront renvoyées sous forme de tampon.

`paramReqList:` ***Corde***

Liste de nombres séparés par des virgules qui représente les options DHCP demandées au serveur par le client. Pour une liste complète des options DHCP, voir https://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml.

`processingTime:` ***Numéro***

Le temps de traitement, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `DHCP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `DHCP.commitRecord` sur l'un ou l'autre `DHCP_REQUEST` ou `DHCP_RESPONSE` événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

DHCP_REQUEST	DHCP_RESPONSE
clientIsExternal	clientIsExternal
clientReqDelay	error
gwAddr	gwAddr
htype	htype
msgType	msgType
receiverIsExternal	offeredAddr
reqBytes	processingTime
reqL2Bytes	rspBytes
reqPkts	rspL2Bytes
senderIsExternal	rspPkts
serverIsExternal	receiverIsExternal
txId	senderIsExternal
	serverIsExternal
	txId

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

Accès uniquement sur `DHCP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

Accès uniquement sur `DHCP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

Accès uniquement sur `DHCP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur `DHCP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, comme les en-têtes.

Accès uniquement sur `DHCP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

Accès uniquement sur `DHCP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`txId:` ***Numéro***

L'identifiant de transaction.

`vendor:` ***Corde***

L'identifiant de classe de fournisseur (VCI) qui indique le fournisseur exécuté sur le client ou le serveur.

`DICOM`

Le DICOM Le cours (Imagerie numérique et DICOM en médecine) vous permet de stocker des métriques et d'accéder aux propriétés sur `DICOM_REQUEST` et `DICOM_RESPONSE` événements.

Évènements

`DICOM_REQUEST`

S'exécute sur chaque demande DICOM traitée par l'équipement.

`DICOM_RESPONSE`

S'exécute sur chaque réponse DICOM traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `DICOM_REQUEST` ou `DICOM_RESPONSE` événement.

L'événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

`findElement(groupTag:` ***Numéro*** `, elementTag:` ***Numéro*** `):` ***Tampon***

Renvoie une mémoire tampon contenant l'élément de données DICOM spécifié par les numéros de groupe et de balise d'élément transmis.

L'élément de données est représenté par une paire ordonnée unique d' entiers qui représentent les numéros de balise de groupe et d'étiquette d'élément. Par exemple, la paire ordonnée « 0008, 0008 » représente l'élément « type d'image ». UN Registre des éléments de données DICOM et les balises définies sont disponibles sur dicom.nema.org.

`groupTag:` ***Numéro***

Le premier nombre de la paire ordonnée unique d'entiers qui représente un élément de données spécifique.

`elementTag:` ***Numéro***

Le deuxième numéro de la paire ordonnée unique ou des entiers qui représentent un élément de données spécifique.

Propriétés

`calledAETitle:` ***Corde***

Titre de l'entité d'application (AE) de l'équipement ou du programme de destination.

`callingAETitle:` ***Corde***

Titre de l'entité d'application (AE) de l'équipement ou du programme source.

`elements:` ***Array***

Ensemble d'éléments de commande et d'éléments de données contenant des valeurs de données de présentation (PDV) constituant un message DICOM.

`error:` ***Corde***

Le message d'erreur détaillé enregistré par le système ExtraHop.

`isReqAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la demande DICOM ne soit terminée.

Accès uniquement sur `DICOM_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`isRspAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la réponse DICOM ne soit terminée.

Accès uniquement sur `DICOM_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`isSubOperation:` ***Booléen***

La valeur est `true` si la métrique de synchronisation sur un L7 le message du protocole n'est pas disponible car la demande ou la réponse principale n' est pas complète.

`methods:` ***Tableau de chaînes***

Tableau de champs de commande dans le message. Chaque champ de commande spécifie un nom d' opération DIMSE, tel que N-CREATE-RSP.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n' est pas valide.

Accès uniquement sur `DICOM_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `DICOM.commitRecord` sur l'un ou l'autre `DICOM_REQUEST` ou `DICOM_RESPONSE` événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

DICOM_REQUEST	DICOM_RESPONSE
calledAETitle	calledAETitle
callingAETitle	callingAETitle
clientIsExternal	clientIsExternal
clientZeroWnd	clientZeroWnd
error	error
isReqAborted	isRspAborted
isSubOperation	isSubOperation
method	method
receiverIsExternal	processingTime
reqPDU	receiverIsExternal
reqSize	rspPDU
reqTransferTime	rspSize
senderIsExternal	rspTransferTime
serverIsExternal	senderIsExternal
serverZeroWnd	serverIsExternal
version	serverZeroWnd
	version

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

Accès uniquement sur `DICOM_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

`reqPDU:` ***Corde***

L'unité de données de protocole (PDU), ou format de message, de la demande.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

`reqSize:` ***Numéro***

Le nombre d'octets de requête L7.

Accès uniquement sur `DICOM_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`reqTransferTime:` ***Numéro***

Le temps de transfert de la demande, exprimé en millisecondes.

Accès uniquement sur `DICOM_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur `DICOM_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur `DICOM_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, comme les en-têtes.

`rspPDU:` ***Corde***

L'unité de données de protocole (PDU), ou format de message, de la réponse.

Accès uniquement sur `DICOM_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

`rspSize:` ***Numéro***

Le nombre d'octets de réponse L7.

Accès uniquement sur `DICOM_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspTransferTime:` ***Numéro***

Le temps de transfert de réponse, exprimé en millisecondes.

Accès uniquement sur `DICOM_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`version:` ***Numéro***

Le numéro de version du DICOM.

`DNS`

Le DNS la classe vous permet de stocker des métriques et d'accéder à des propriétés sur `DNS_REQUEST` et `DNS_RESPONSE` événements.

Évènements

`DNS_REQUEST`

S'exécute sur chaque requête DNS traitée par l'équipement.

`DNS_RESPONSE`

S'exécute sur chaque réponse DNS traitée par l'équipement.

Méthodes

`answersInclude(term:` ***Corde*** | ***Adresse IP*** `):` ***Booléen***

Retours `true` si le terme spécifié est présent dans une réponse DNS. Pour les termes de chaîne, la méthode vérifie à la fois le nom et l'enregistrement de données dans la section des réponses de la réponse. Pour IPAddress termes, la méthode vérifie uniquement l'enregistrement des données dans la section des réponses.

Ne peut être appelé que sur DNS_RESPONSE événements.

`commitRecord()`: ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un DNS_REQUEST ou DNS_RESPONSE événement.

L'événement détermine quelles propriétés sont validées pour l'objet d'enregistrement. Pour afficher les propriétés par défaut validées pour chaque événement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`answers`: ***Array***

Tableau d'objets correspondant aux enregistrements de ressources de réponses.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

Les objets contiennent les propriétés suivantes :

`data`: ***Corde*** | ***Adresse IP***

La valeur des données dépend du type. La valeur est `null` pour les types d'enregistrement non pris en charge. Les types d'enregistrement pris en charge incluent :

- A
- AAAA
- NS
- PTR
- CNAME
- MX
- SRV
- SOA
- TXT

`name`: ***Corde***

Le nom de l'enregistrement.

`ttl`: ***Numéro***

La valeur de la durée de vie.

`type`: ***Corde***

Type d'enregistrement DNS.

`typeNum`: ***Numéro***

Représentation numérique du type d'enregistrement DNS.

`error`: ***Corde***

Le nom du code d'erreur DNS, conformément aux paramètres DNS de l'IANA.

Renvoie OTHER pour les codes d'erreur qui ne sont pas reconnus par le système ; toutefois, `errorNum` spécifie la valeur du code numérique.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

`errorNum`: ***Numéro***

Représentation numérique du code d'erreur DNS conformément aux paramètres DNS de l'IANA.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

`isAuthenticData`: ***Booléen***

La valeur est `true` si la réponse a été validée par DNSSEC.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

`isAuthoritative:` ***Booléen***

La valeur est `true` si la réponse faisant autorité est définie dans la réponse.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

`isCheckingDisabled:` ***Booléen***

La valeur est `true` si une réponse doit être renvoyée même si la demande n'a pas pu être authentifiée.

Accès uniquement sur DNS_REQUEST événements ; sinon, une erreur se produira.

`isDGADomain:` ***Booléen***

La valeur est `true` si le domaine du serveur a peut-être été généré par un algorithme de génération de domaine (DGA). Certaines formes de programmes malveillants produisent un grand nombre de noms de domaine dotés de DGA pour masquer les serveurs de commande et de contrôle. La valeur est `null` si le domaine n'était pas suspect.

`isRecursionAvailable:` ***Booléen***

La valeur est `true` si le serveur de noms prend en charge les requêtes récursives.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

`isRecursionDesired:` ***Booléen***

La valeur est `true` si le serveur de noms doit exécuter la requête de manière récursive.

Accès uniquement sur DNS_REQUEST événements ; sinon, une erreur se produira.

`isReqTimeout:` ***Booléen***

La valeur est `true` si le délai de la demande a expiré.

Accès uniquement sur DNS_REQUEST événements ; sinon, une erreur se produira.

`isRspTruncated:` ***Booléen***

La valeur est `true` si la réponse est tronquée.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

`opcode:` ***Corde***

Le nom du code d'opération DNS conformément aux paramètres DNS de l'IANA. Les codes suivants sont reconnus par le système ExtraHop :

Code d'opération	Nom
0	Query
1	IQuery (Inverse Query - Obsolete)
2	Status
3	Unassigned
4	Notify
5	Update
6-15	Unassigned

Renvoie OTHER pour les codes qui ne sont pas reconnus par le système ; toutefois, le `opcodeNum` La propriété spécifie la valeur du code numérique.

`opcodeNum:` ***Numéro***

Représentation numérique du code d'opération DNS conformément aux paramètres DNS de l'IANA.

`payload:` ***Tampon***

Le Tampon objet qui contient les octets de charge utile bruts de la transaction d'événement.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en octets. La valeur est `NaN` en cas de réponses mal formées et abandonnées ou si le timing n' est pas valide.

Accès uniquement sur `DNS_RESPONSE` événements ; sinon, une erreur se produira.

`qname:` ***Corde*** | ***nul***

Le nom d'hôte demandé.

Cette valeur est `null` si le `opcode` la propriété est `UPDATE`.

`qtype:` ***Corde*** | ***nul***

Le nom du type d'enregistrement de demande DNS conformément aux paramètres DNS de l'IANA.

Retours `OTHER` pour les types qui ne sont pas reconnus par le système ; toutefois, le `qtypeNum` La propriété spécifie la valeur de type numérique.

Cette valeur est `null` si le `opcode` la propriété est `UPDATE`.

`qtypeNum:` ***Numéro*** | ***nul***

Représentation numérique du type d'enregistrement de demande DNS conformément aux paramètres DNS de l'IANA.

Cette valeur est `null` si le `opcode` la propriété est `UPDATE`.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `DNS.commitRecord()` sur l'un ou l'autre `DNS_REQUEST` ou `DNS_RESPONSE` événement.

L'événement pour lequel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

DNS_REQUEST	DNS_RESPONSE
clientIsExternal	answers
clientZeroWnd	clientIsExternal
isCheckingDisabled	clientZeroWnd
isDGADomain	error
isRecursionDesired	isAuthoritative
isReqTimeout	isCheckingDisabled
opcode	isDGADomain
qname	isRecursionAvailable
qtype	isRspTruncated
receiverIsExternal	opcode
reqBytes	processingTime
reqL2Bytes	receiverIsExternal
reqPkts	qname
senderIsExternal	qtype
serverIsExternal	rspBytes
serverZeroWnd	rspL2Bytes
	rspPkts

DNS_REQUEST	DNS_RESPONSE
	senderIsExternal
	serverIsExternal
	serverZeroWnd

reqBytes: ***Numéro***

Le nombre de L4 octets de requête, à l'exclusion des en-têtes L4.

Accès uniquement sur DNS_REQUEST événements ; sinon, une erreur se produira.

reqL2Bytes: ***Numéro***

Le nombre de L2 octets de requête, y compris les en-têtes L2.

Accès uniquement sur DNS_REQUEST événements ; sinon, une erreur se produira.

reqPkts: ***Numéro***

Le nombre de paquets de requêtes.

Accès uniquement sur DNS_REQUEST événements ; sinon, une erreur se produira.

rspBytes: ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

reqZeroWnd: ***Numéro***

Le nombre de fenêtres nulles dans la demande.

rspL2Bytes: ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

rspPkts: ***Numéro***

Nombre d'octets de réponse au niveau de l'application.

Accès uniquement sur DNS_RESPONSE événements ; sinon, une erreur se produira.

rspZeroWnd: ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

txId: ***Numéro***

L'ID de transaction de la demande ou de la réponse DNS.

zname: ***Corde*** | ***nul***

La zone DNS en cours de mise à jour.

Cette valeur est null si le opcode la propriété n'est pas UPDATE.

ztype: ***Corde*** | ***nul***

Type de zone DNS en cours de mise à jour. Retours OTHER pour les types qui ne sont pas reconnus par le système.

Cette valeur est null si le opcode la propriété n'est pas UPDATE.

ztypeNum: ***Numéro*** | ***nul***

Représentation numérique du type de zone DNS.

Cette valeur est null si le opcode la propriété n'est pas UPDATE.

FIX

Le FIX la classe vous permet de stocker des métriques et d'accéder aux propriétés sur FIX_REQUEST et FIX_RESPONSE événements.

Évènements

FIX_REQUEST

S'exécute sur chaque demande FIX traitée par l'équipement.

FIX_RESPONSE

S'exécute sur chaque réponse FIX traitée par l'équipement.

Note: Le FIX_RESPONSE l'événement correspond à une demande basée sur le numéro de commande. Il n'y a pas de corrélation univoque entre la demande et la réponse. Il peut y avoir des demandes sans réponse, et parfois les données sont transmises au client, qui limite la disponibilité des données de demande lors d'un événement de réponse. Cependant, vous pouvez invoquer la table de session pour résoudre des scénarios complexes tels que l'identifiant de commande de soumission.

Méthodes

commitRecord(): ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un FIX_REQUEST ou FIX_RESPONSE événement.

L'événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le record propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si commitRecord() méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

fields: ***Array***

Une liste de champs FIX. Comme ils sont basés sur du texte, les champs du protocole clé-valeur sont exposés sous la forme d'un tableau d'objets dont les propriétés de nom et de valeur contiennent des chaînes. Par exemple :

```
8=FIX.4.2<SOH>9=233<SOH>35=G<SOH>34=206657...
```

se traduit par :

```
{"BeginString": "FIX.4.2", "BodyLength": "233", "MsgType": "G",
 "MsgSeqNum":
"206657"}
```

La représentation des chaînes clés est traduite, si possible. Pour les extensions, une représentation numérique est utilisée. Par exemple, il n'est pas possible de déterminer 9178=0 (comme on le voit dans les captures réelles). La clé est plutôt traduite en « 9178 ». Les champs sont extraits après la longueur du message et les champs de version sont extraits jusqu'à la somme de contrôle (dernier champ). La somme de contrôle n'est pas extraite.

Dans l'exemple suivant, le déclencheur debug(JSON.stringify(FIX.fields)); affiche les champs suivants :

```
[
```

```
    {"name":"MsgType","value":"0"},
    {"name":"MsgSeqNum","value":"2"},
    {"name":"SenderCompID","value":"AA"},
    {"name":"SendingTime","value":"20140904-03:49:58.600"},
    {"name":"TargetCompID","value":"GG"}
]
```

Pour déboguer et imprimer tous les champs FIX, activez le débogage sur le déclencheur et entrez le code suivant :

```
var fields = '';
for (var i = 0; i < FIX.fields.length; i++) {
fields += '"' + FIX.fields[i].name + '" : "' + FIX.fields[i].value +
'"\n';
} debug(fields);
```

Le résultat suivant est affiché dans le journal de débogage du déclencheur :

```
"MsgType" : "5"
"MsgSeqNum" : "3"
"SenderCompID" : "GRAPE"
"SendingTime" : "20140905-00:10:23.814"
"TargetCompID" : "APPLE"
```

`msgType:` ***Corde***

La valeur de la clé MessageCompid.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. La valeur est `NaN` si le chronométrage n'est pas valide.

Accès uniquement sur `CORRIGE_RÉPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `FIX.commitRecord` sur l'un ou l'autre `FIX_REQUEST` ou `FIX_RESPONSE` événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

FIX_REQUEST	FIX_RESPONSE
clientIsExternal	clientIsExternal
clientZeroWnd	clientZeroWnd
msgType	msgType
receiverIsExternal	receiverIsExternal
reqBytes	rspBytes
reqL2Bytes	rspL2Bytes
reqPkts	rspPkts
reqRTO	rspRTO
sender	sender
senderIsExternal	senderIsExternal

FIX_REQUEST	FIX_RESPONSE
serverIsExternal	serverIsExternal
serverZeroWnd	serverZeroWnd
target	target
version	version

reqBytes: ***Numéro***
Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

reqL2Bytes: ***Numéro***
Le nombre de L2 octets de demande, y compris les en-têtes L2.

reqPkts: ***Numéro***
Le nombre de paquets de demandes.

reqRTO: ***Numéro***
Le numéro de demande délais de retransmission (RTO).

reqZeroWnd: ***Numéro***
Le nombre de fenêtres nulles dans la demande.

rspBytes: ***Numéro***
Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

rspL2Bytes: ***Numéro***
Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

rspPkts: ***Numéro***
Le nombre de paquets de réponse.

rspRTO: ***Numéro***
Le nombre de réponses délais de retransmission (RTO).

rspZeroWnd: ***Numéro***
Le nombre de fenêtres nulles dans la réponse.

sender: ***Corde***
La valeur de la clé SenderCompID.

target: ***Corde***
La valeur de la clé TargetCompID.

version: ***Corde***
Version du protocole.

FTP

Le FTP la classe vous permet de stocker des métriques et d'accéder à des propriétés sur FTP_REQUEST et FTP_RESPONSE événements.

Évènements

FTP_REQUEST
S'exécute sur chaque requête FTP traitée par l'équipement.

FTP_RESPONSE
S'exécute sur chaque réponse FTP traitée par l'équipement.

Méthodes

`commitRecord()` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `FTP_RESPONSE` événement. Enregistrez les validations le `FTP_REQUEST` les événements ne sont pas pris en charge.

Pour afficher les propriétés par défaut validées pour l' objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`args:` ***Corde***

Les arguments de la commande.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`cwd:` ***Corde***

Dans le cas d'un utilisateur de `/`, lorsque client envoie un « subdir CWD » :

- La valeur est `/` quand method == « CWD ».
- La valeur est `/subdir` pour les commandes suivantes (plutôt que CWD ne devienne le répertoire modifié dans le cadre du déclencheur de réponse CWD).

Inclut «... » au début du chemin en cas d'événement de resynchronisation ou de tronquage du chemin .

Inclut «... » à la fin du chemin si celui-ci est trop long. Le chemin est tronqué à 4 096 caractères.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`error:` ***chaîne***

Message d'erreur détaillé enregistré par le système ExtraHop.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`filename:` ***Corde***

Le nom du fichier en cours de transfert.

`isReqAborted:` ***Booléen***

La valeur est `true` la connexion est fermée avant que la requête FTP ne soit terminée.

`isRspAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant la fin de la réponse FTP.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`method:` ***Corde***

La méthode FTP.

`path:` ***Corde***

Le chemin des commandes FTP. Inclut «... » au début du chemin en cas d'événement de resynchronisation ou de tronquage du chemin. Inclut «... » à la fin du chemin si celui-ci est trop long. Le chemin est tronqué à 4 096 caractères.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`payloadMediaType:` ***Corde | Null***

Type de support contenu dans la charge utile. La valeur est nulle s'il n'y a aucune charge utile ou si le type de support est inconnu.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes (équivalent à `rspTimeToFirstPayload - reqTimeToLastByte`). La valeur est `NaN` en cas de réponses mal formées et abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `FTP.commitRecord()` sur un `FTP_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `args`
- `clientIsExternal`
- `clientZeroWnd`
- `cwd`
- `error`
- `isReqAborted`
- `isRspAborted`
- `method`
- `path`
- `processingTime`
- `receiverIsExternal`
- `reqBytes`
- `reqL2Bytes`
- `reqPayloadMediaType`
- `reqPayloadSHA256`
- `reqPkts`
- `reqRTO`
- `roundTripTime`
- `rspBytes`
- `rspL2Bytes`
- `rspPayloadMediaType`
- `rspPayloadSHA256`
- `rspPkts`
- `rspRTO`
- `senderIsExternal`
- `serverIsExternal`
- `serverZeroWnd`
- `statusCode`
- `transferBytes`
- `user`

Accédez à l'objet d'enregistrement uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`reqBytes:` ***Numéro***

Le nombre de L4 octets de requête, à l'exclusion des en-têtes L4.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de requête, y compris les en-têtes L2.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`reqPkts:` ***Numéro***

Le nombre de paquets de requêtes.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Le temps médian aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`statusCode:` ***Numéro***

Le code dveloppe l'état FTP de la réponse.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

Les codes suivants sont valides :

Code	Descriptif
110	Redémarrez Marker Replay.
120	Service prêt en *nnn* minutes.
125	La connexion de données est déjà ouverte ; le transfert commence.
150	État du fichier correct ; connexion de données sur le point d'ouvrir.
202	Commande non implémentée, superflue sur ce site.
211	État du système ou réponse d'aide du système.
212	État du répertoire.
213	État du fichier.

Code	Descriptif
214	Message d'aide.
215	Type de système NAME.
220	Service prêt pour les nouveaux utilisateurs.
221	Connexion de commande de fermeture du service.
225	Connexion de données ouverte ; aucun transfert en cours.
226	Fermeture de la connexion de données. L'action demandée sur le fichier a réussi.
227	Entrée en mode passif.
228	Entrée en mode passif long.
229	Entrée en mode passif étendu.
230	Utilisateur connecté, continuez. Déconnecté le cas échéant.
231	L'utilisateur s'est déconnecté ; le service a été interrompu.
232	Commande de déconnexion notée, elle sera terminée une fois le transfert terminé
250	L'action demandée sur le fichier est correcte, terminée.
257	« PATHNAME » a été créé.
331	Nom d'utilisateur correct, mot de passe requis.
332	Vous avez besoin d'un compte pour vous connecter.
350	Action sur le fichier demandée dans l'attente de plus amples informations.
421	Service non disponible, fermeture de la connexion de commande.
425	Impossible d'ouvrir la connexion de données.
426	Connexion fermée ; transfert interrompu.
430	Nom d'utilisateur ou mot de passe non valide.
434	L'hôte demandé n'est pas disponible.
450	L'action demandée sur le fichier n'a pas été effectuée.
451	L'action demandée a été abandonnée. Erreur locale lors du traitement.
452	L'action demandée n'a pas été prise.
501	Erreur de syntaxe dans les paramètres ou les arguments.
502	Commande non implémentée.
503	Mauvaise séquence de commandes.
504	Commande non implémentée pour ce paramètre.
530	Non connecté.
532	Vous avez besoin d'un compte pour stocker des fichiers.
550	L'action demandée n'a pas été prise. Le fichier n'est pas disponible.

Code	Descriptif
551	L'action demandée a été abandonnée. Type de page inconnu.
552	L'action demandée sur le fichier a été abandonnée. Allocation de stockage dépassée.
553	L'action demandée n'a pas été prise. Nom de fichier non autorisé.
631	Réponse protégée par intégrité.
632	Réponse protégée en termes de confidentialité et d'intégrité.
633	Réponse protégée en toute confidentialité.
10054	Réinitialisation de la connexion par un pair.
10060	Impossible de se connecter au serveur distant.
10061	Impossible de se connecter au serveur distant. La connexion est active refusée.
10066	Le répertoire n'est pas vide.
10068	Trop d'utilisateurs, le serveur est plein.

`transferBytes:` ***Numéro***

Le nombre d'octets transférés sur le canal de données au cours d'une `FTP_RESPONSE` événement.

Accès uniquement sur `FTP_RESPONSE` événements ; sinon, une erreur se produira.

`user:` ***Corde***

Le nom d'utilisateur, s'il est disponible. Dans certains cas, par exemple lorsque les événements de connexion sont chiffrés, le nom d'utilisateur n'est pas disponible.

`HL7`

Le HL7 la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `HL7_REQUEST` et `HL7_RESPONSE` événements.

Évènements

`HL7_REQUEST`

Fonctionne sur chaque demande HL7 traitée par l'équipement.

`HL7_RESPONSE`

Fonctionne sur chaque réponse HL7 traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `HL7_RESPONSE` événement. Enregistrer les validations sur `HL7_REQUEST` les événements ne sont pas pris en charge.

Pour consulter les propriétés par défaut attribuées à l' objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`ackCode:` ***Corde***

Le code de confirmation à deux caractères.

Accès uniquement sur `HL7_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`ackId:` ***Corde***

Identifiant du message accusé de réception.

Accès uniquement sur `HL7_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`msgId:` ***Corde***

Identifiant unique de ce message.

`msgType:` ***Corde***

Le champ complet du type de message, y compris le sous-champ MsgID.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n' est pas valide.

Accès uniquement sur `HL7_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `HL7.commitRecord()` sur un `HL7_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `ackCode`
- `ackId`
- `clientIsExternal`
- `clientZeroWnd`
- `msgId`
- `msgType`
- `receiverIsExternal`
- `roundTripTime`
- `processingTime`
- `senderIsExternal`
- `serverIsExternal`
- `serverZeroWnd`
- `version`

Accédez à l'objet d'enregistrement uniquement sur `HL7_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur `HL7_RESPONSE` événements ; sinon, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`segments:` ***Array***

Un tableau d'objets segmentés contenant les champs suivants :

`name:` ***Corde***

Le nom du segment.

`fields:` ***Tableau de chaînes***

Les valeurs des champs du segment. Comme les indices du tableau commencent à 0 et que les numéros de champ HL7 commencent à 1, l'index est le numéro de champ HL7 moins 1. Par exemple, pour sélectionner le champ 16 d'un segment PRT (l'identifiant de l'équipement de participation), spécifiez 15, comme indiqué dans l'exemple de code suivant :

```
HL7.segments[5].fields[15]
```

Note: Si un segment est vide, le tableau contient une chaîne vide à l'index du segment.

`subfieldDelimiter:` ***Corde***

Supporte les délimiteurs de champs non standard.

`version:` ***Corde***

Version annoncée dans le segment MSH.

Note: La quantité de données mises en mémoire tampon est limitée par l'option de capture suivante : `("message_length_max": number)`

`HTTP`

Le HTTP la classe vous permet de stocker des métriques et d'accéder à des propriétés sur `HTTP_REQUEST` et `HTTP_RESPONSE` événements.

Évènements

`HTTP_REQUEST`

S'exécute sur chaque requête HTTP traitée par l'équipement.

`HTTP_RESPONSE`

S'exécute sur chaque réponse HTTP traitée par l'équipement.

Des options de charge utile supplémentaires sont disponibles lorsque vous créez un déclencheur qui s'exécute sur l'un de ces événements. Voir Options de déclencheur avancées pour plus d' informations.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `HTTP_REQUEST` ou `HTTP_RESPONSE` événement. Pour afficher les propriétés par défaut validées pour l'objet d' enregistrement, consultez `record` propriété ci-dessous.

Si le `commitRecord()` méthode est appelée `HTTP_REQUEST` événement, l'enregistrement n'est créé que lorsque `HTTP_RESPONSE` l'événement se déroule. Si le `commitRecord()` la méthode est appelée à la fois sur `HTTP_REQUEST` et le correspondant `HTTP_RESPONSE`, un seul enregistrement est créé pour la demande et la réponse, même si `commitRecord()` La méthode est appelée plusieurs fois lors des mêmes événements déclencheurs.

`findHeaders(name:` ***Corde*** `):` ***Array***

Permet d'accéder aux valeurs d'en-tête HTTP et renvoie un tableau d'objets d'en-tête (avec des propriétés de nom et de valeur) dont les noms correspondent au préfixe de la valeur de chaîne. Voir Exemple : accéder aux attributs d'en-tête HTTP pour plus d'informations.

`parseQuery(String):` ***Objet***

Accepte une chaîne de requête et renvoie un objet dont les noms et les valeurs correspondent à ceux de la chaîne de requête, comme illustré dans l' exemple suivant :

```
var query = HTTP.parseQuery(HTTP.query);
```

```
debug("user id: " + query.userid);
```

Note: Si la chaîne de requête contient des clés répétées, les valeurs correspondantes sont renvoyées dans un tableau. Par exemple, la chaîne de requête `event_type=status_update_event&event_type=api_post_event` renvoie l' objet suivant :

```
{
  "event_type": ["status_update_event", "api_post_event"]
}
```

Propriétés

`age:` ***Numéro***

Pour `HTTP_REQUEST` événements, le temps écoulé entre le premier octet de la demande et le dernier octet vu de la demande. Pour `HTTP_RESPONSE` events, le temps écoulé entre le premier octet de la demande et le dernier octet vu de la réponse. Le temps est exprimé en millisecondes. Spécifie une valeur valide pour les demandes mal formées et abandonnées. La valeur est `NaN` en cas de demandes et de réponses expirées, ou si le timing n'est pas valide.

`contentType:` ***Corde***

La valeur de l'en-tête HTTP du type de contenu.

`cookies:` ***Array***

Tableau d'objets représentant les cookies et contenant des propriétés telles que « domaine » et « expire ». Les propriétés correspondent aux attributs de chaque cookie, comme indiqué dans l' exemple suivant :

```
var cookies = HTTP.cookies,
    cookie,
    i;
for (i = 0; i < cookies.length; i++) {
    cookie = cookies[i];
    if (cookie.domain) {
        debug("domain: " + cookie.domain);
    }
}
```

`encryptionProtocol:` ***Corde***

Le protocole avec lequel la transaction est cryptée.

`filename:` ***Corde | Null***

Le nom du fichier en cours de transfert. Si la requête ou la réponse HTTP n'a pas transféré de fichier, la valeur est nulle.

`headers:` ***Objet***

Un objet semblable à un tableau qui permet d'accéder aux noms et valeurs des en-têtes HTTP. Les informations d'en-tête sont disponibles via l'une des propriétés suivantes :

`length:` ***Numéro***

Le nombre d'en-têtes.

`string property:`

Le nom de l'en-tête, accessible à la manière d'un dictionnaire, comme illustré dans l'exemple suivant :

```
var headers = HTTP.headers;
    session = headers["X-Session-Id"];
    accept = headers.accept;
```

`numeric property:`

Correspond à l'ordre dans lequel les en-têtes apparaissent sur le fil. L' objet renvoyé possède un nom et une propriété value. Les propriétés numériques sont utiles pour effectuer des itérations sur tous les en-têtes et lever l'ambiguïté des en-têtes dont le nom est dupliqué, comme illustré dans l'exemple suivant :

```
var headers = HTTP.headers;
for (i = 0; i < headers.length; i++) {
    hdr = headers[i];
    debug("headers[" + i + "].name: " + hdr.name);
    debug("headers[" + i + "].value: " + hdr.value);
}
```

Note: Épargner `HTTP.headers` dans le magasin Flow n'enregistre pas toutes les valeurs d'en-tête individuelles. Il est recommandé d'enregistrer les valeurs d'en-tête individuelles dans le magasin Flow. Reportez-vous à Flow section de classe pour plus de détails.

`headersRaw:` ***Corde***

Le bloc non modifié d'en-têtes HTTP, exprimé sous forme de chaîne.

`host:` ***Corde***

La valeur de l'en-tête de l'hôte HTTP.

`isClientReset:` ***Booléen***

La valeur est `true` si le flux HTTP/2 est réinitialisé par le client. Si le protocole est HTTP1.1, la valeur est `false`.

`isContinued:` ***Booléen***

La valeur est `true` si le client a envoyé une requête HTTP/1.1 initiale avec un `Expect: 100-continue` en-tête et a reçu un code dquo 100 du serveur dans le cadre de la transaction. Si le protocole est HTTP/2, la valeur est `false`

`isDesync:` ***Booléen***

La valeur est `true` si l'analyseur de protocole a été désynchronisé en raison de paquets manquants.

`isEncrypted:` ***Booléen***

La valeur est `true` si la transaction est effectuée via HTTP sécurisé.

`isDecrypted:` ***Booléen***

La valeur est `true` si le système ExtraHop a déchiffré et analysé la transaction de manière sécurisée. L'analyse du trafic déchiffré peut révéler les menaces avancées qui se cachent dans le trafic chiffré.

`isPipelined:` ***Booléen***

La valeur est `true` si la transaction est en pipeline.

`isReqAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant la fin de la requête HTTP.

`isRspAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant la fin de la réponse HTTP.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`isRspChunked:` ***Booléen***

La valeur est `true` si la réponse est fragmentée.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`isRspCompressed:` ***Booléen***

La valeur est `true` si la réponse est compressée.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`isServerPush:` ***Booléen***

La valeur est `true` si la transaction est le résultat d'un push du serveur.

`isServerReset:` ***Booléen***

La valeur est `true` si le flux HTTP/2 est réinitialisé par le serveur.

`isSQLi:` ***Booléen***

La valeur est vraie si la demande comprenait un ou plusieurs fragments SQL suspects. Ces fragments indiquent une injection SQL potentielle (SQLi). SQLi est une technique qui permet à un attaquant d'accéder à des données et de les modifier en insérant des instructions SQL malveillantes dans une requête SQL.

`isXSS:` ***Booléen***

La valeur est vraie si la requête HTTP incluait des tentatives potentielles de script intersite (XSS). Une tentative XSS réussie peut injecter un script ou une charge utile malveillant côté client dans un site Web ou une application de confiance. Lorsqu'une victime visite le site Web, le script malveillant est ensuite injecté dans le navigateur de la victime.

`method:` ***Corde***

La méthode HTTP de la transaction, telle que POST et GET.

`origin:` ***Adresse IP | Corde***

La valeur de l'en-tête X-Forwarded-For ou true-client-ip.

`path:` ***Corde***

La partie chemin de l'URI : `/path/`.

`payload:` ***Tampon | Null***

Le Tampon objet qui contient les octets de charge utile bruts de la transaction d'événement. Si la charge utile a été compressée, le contenu décompressé est renvoyé.

La mémoire tampon contient *N* premiers octets de la charge utile, où *N* est le nombre d'octets de charge utile spécifié par le Octets vers la mémoire tampon champ lorsque le déclencheur a été configuré via l'interface utilisateur Web ExtraHop. Le nombre d'octets par défaut est de 2 048. Pour plus d'informations, voir Options de déclencheur avancées.

Le script suivant est un exemple d' analyse de charge utile HTTP :

```
// Extract the user name based on a pattern "user=*&" from payload
// of a login URI that has "auth/login" as a URI substring.

if (HTTP.payload && /auth\/login/i.test(HTTP.uri)) {
    var user = /user=(.*?)\&/i.exec(HTTP.payload);
    if (user !== null) {
        debug("user: " + user[1]);
    }
}
```

Note: Si deux déclencheurs de mise en mémoire tampon de charge utile HTTP sont attribués au même équipement, la valeur la plus élevée est sélectionnée et la valeur de `HTTP.payload` est identique pour les deux déclencheurs.

`payloadParts:` ***Tableau d'objets | Null***

Tableau d'objets contenant les charges utiles individuelles d'une requête ou d'une réponse HTTP en plusieurs parties. La valeur est nulle si le type de contenu n'est pas multipart. Chaque objet contient les champs suivants :

`headers:` ***Objet***

Un objet d'en-tête qui spécifie les en-têtes HTTP. Pour plus d'informations, consultez la description du `HTTP.headers` propriété.

`payloadSHA256:` ***Corde***

Représentation hexadécimale du hachage SHA-256 de la charge utile. La chaîne ne contient aucun délimiteur.

`payloadMediaType:` ***Corde | Null***

Type de support de la charge utile. La valeur est nulle si le type de média est inconnu.

`payload:` ***Tampon***

Le Tampon objet contenant les octets de charge utile bruts.

`size:` ***Numéro***

Taille de la charge utile, exprimée en octets.

`filename:` ***Corde***

Le nom de fichier spécifié dans l'en-tête Content-Disposition.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes (équivalent à `rspTimeToFirstPayload - reqTimeToLastByte`). La valeur est `NaN` en cas de réponses mal formées et abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `HTTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`query:` ***Corde***

La partie de chaîne de requête du `URI: query=string`. Elle suit généralement l'URL et est séparée de celle-ci par un point d'interrogation. Les chaînes de requête multiples sont séparées par une esperluette (&) ou un point-virgule (;)).

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `HTTP.commitRecord()`.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `clientZeroWnd`
- `contentType`
- `host`
- `isPipelined`
- `isReqAborted`
- `isRspAborted`
- `isRspChunked`
- `isRspCompressed`
- `method`
- `oauthAlgorithm`
- `oauthAudience`
- `oauthClientId`
- `oauthIssuer`
- `origin`
- `query`
- `receiverIsExternal`
- `referer`
- `reqBytes`
- `reqL2Bytes`
- `reqPayloadMediaType`
- `reqPayloadSHA256`
- `reqPkts`
- `reqRTO`

- reqSize
- reqTimeToLastByte
- roundTripTime
- rspBytes
- rspL2Bytes
- rspPayloadMediaType
- rspPayloadSHA256
- rspPkts
- rspRTO
- rspSize
- rspTimeToFirstHeader
- rspTimeToFirstPayload
- rspTimeToLastByte
- rspVersion
- samlRspAudience
- samlRspCertificateSubject
- samlRspDigestMethodAlgorithm
- samlRspIssuer
- samlRspNameID
- samlRspSignatureMethodAlgorithm
- samlRspStatusCode
- senderIsExternal
- serverIsExternal
- serverZeroWnd
- statusCode
- thinkTime
- title
- processingTime
- uri
- userAgent

Accédez à l'objet d'enregistrement uniquement sur HTTP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`referer:` ***Corde***

La valeur de l'en-tête du référent HTTP.

`reqBytes:` ***Numéro***

Le nombre de L4 octets de requête, à l'exclusion des en-têtes L4.

Accès uniquement sur HTTP_RESPONSE événements ; sinon, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de requête, y compris les en-têtes L2.

Accès uniquement sur HTTP_RESPONSE événements ; sinon, une erreur se produira.

`reqPkts:` ***Numéro***

Le nombre de paquets de requêtes.

Accès uniquement sur HTTP_RESPONSE événements ; sinon, une erreur se produira.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

Accès uniquement sur HTTP_RESPONSE événements ; sinon, une erreur se produira.

`reqSize:` ***Numéro***

Nombre d'octets de requête L7, à l'exclusion des en-têtes HTTP.

`reqTimeToLastByte:` ***Numéro***

Le temps écoulé entre le premier octet de la demande et le dernier octet de la demande, exprimé en millisecondes. La valeur est `NaN` en cas de demandes et de réponses expirées, ou si le timing n'est pas valide.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Temps d'aller-retour (RTT) TCP médian, exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspSize:` ***Numéro***

Le nombre d'octets de réponse L7, à l'exclusion des en-têtes HTTP.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspTimeToFirstHeader:` ***Numéro***

Temps écoulé entre le premier octet de la demande et la ligne d'état qui précède les en-têtes de réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspTimeToFirstPayload:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le premier octet de charge utile de la réponse, exprimé en millisecondes. Renvoie une valeur nulle lorsque la réponse ne contient pas de charge utile. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur `HTTP_RESPONSE` événements ; sinon, une erreur se produira.

`rspVersion:` ***Corde***

Version HTTP de la réponse.

Accès uniquement sur `HTTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`samlRequestXML:` ***Tampon | Null***

Le Tampon objet contenant les octets XML bruts de la requête SAML. Si la requête ou la réponse HTTP ne contenait pas de requête SAML, la valeur est `null`.

`samlResponseXML:` ***Tampon | Null***

Le Tampon objet contenant les octets XML bruts de la réponse SAML. Si la requête ou la réponse HTTP ne contenait pas de réponse SAML, la valeur est `null`.

`sqli:` ***Tableau de cordes***

Tableau de fragments SQL suspects inclus dans la demande. Ces fragments peuvent contenir une injection SQL potentielle (SQLi). SQLi est une technique qui permet à un attaquant d' accéder à des données et de les modifier en insérant des instructions SQL malveillantes dans une requête SQL.

`statusCode:` ***Numéro***

Le code dquo HTTP de la réponse.

Accès uniquement sur `HTTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

Note: Renvoie un code dcoagulant 0 s'il n'est pas valide `HTTP_RESPONSE` est reçu.

`streamId:` ***Numéro***

L'ID du flux qui a transféré la ressource. Les réponses pouvant être renvoyées dans l'ordre, cette propriété est requise pour que les transactions HTTP/2 fassent correspondre les demandes aux réponses. La valeur est `1` pour la demande de mise à niveau HTTP/1.1 et `null` pour les versions HTTP précédentes.

`title:` ***Corde***

La valeur de l'élément de titre du contenu HTML, s'il est présent. Si le titre a été compressé, le contenu décompressé est renvoyé.

`thinkTime:` ***Numéro***

Le temps écoulé entre le moment où le serveur a transféré la réponse au client et le client qui transfère une nouvelle demande au serveur, exprimée en millisecondes. La valeur est `NaN` s'il n'y a pas de mesure valide.

`uri:` ***Corde***

L'URI sans chaîne de requête : `f.q.d.n/path/`.

`userAgent:` ***Corde***

La valeur de l'en-tête HTTP user-agent.

`xss:` ***Tableau de cordes***

Tableau de fragments de requêtes HTTP suspects inclus dans la requête. Ces fragments peuvent injecter un script ou une charge utile malveillant côté client dans un site Web ou une application de confiance. Lorsqu'une victime visite le site Web, le script malveillant est ensuite injecté dans le navigateur de la victime.

Exemples de déclencheurs

- Exemple : suivi des réponses HTTP de niveau 500 par ID client et URI
- Exemple : suivre les requêtes SOAP
- Exemple : accéder aux attributs d'en-tête HTTP
- Exemple : enregistrer des données dans une table de session
- Exemple : création d'un conteneur d'applications

IBMMQ

Le IBMMQ la classe vous permet de stocker des métriques et d'accéder aux propriétés sur IBMMQ_REQUEST et IBMMQ_RESPONSE événements.

Note: Le protocole IBMMQ prend en charge le codage EBCDIC.

Évènements

IBMMQ_REQUEST

S'exécute sur chaque demande IBMMQ traitée par l'équipement.

IBMMQ_RESPONSE

S'exécute sur chaque réponse IBMMQ traitée par l'équipement.

Méthodes

commitRecord(): ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un IBMMQ_REQUEST ou IBMMQ_RESPONSE événement.

L'événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le record propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si le commitRecord() méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

channel: ***Corde***

Le nom du canal de communication.

conversationId: ***Numéro***

Identifiant de la conversation MQ.

correlationId: ***Corde***

L'ID de corrélation IBMMQ.

error: ***Corde***

Chaîne d'erreur correspondant au code d'erreur sur le fil.

method: ***Corde***

Nom de la demande de protocole filaire ou de la méthode de réponse.

Les noms des méthodes ExtraHop suivants diffèrent des noms des méthodes Wireshark :

Hop supplémentaire	Wireshark
ASYNC_MSG_V7	ASYNC_MESSAGE
MQCLOSEv7	SOCKET_ACTION
MQGETv7	REQUEST_MSGS
MQGETv7_REPLY	NOTIFICATION

msg: ***Tampon***

UN Tampon objet contenant les messages MQPUT, MQPUT1, MQGET_REPLY, ASYNC_MSG_V7 et MESSAGE_DATA.

Les messages de file d'attente supérieurs à 32 Ko peuvent être divisés en plusieurs segments. Un déclencheur est lancé pour chaque segment et seul le premier segment contient un message non nul.

Les données de la mémoire tampon peuvent être converties en chaîne imprimable via le `toString()` fonctionnel ou formaté via des commandes de décompression.

`msgFormat:` ***Corde***

Le format du message.

`msgId:` ***Corde***

L'ID du message IBMMQ.

`pcfError:` ***Corde***

Chaîne d'erreur correspondant au code d'erreur sur le fil pour le canal PCF (Programmable Command Formats).

`pcfMethod:` ***Corde***

Nom de la méthode de demande ou de réponse du protocole filaire pour le canal PCF (Programmable Command Formats).

`pcfWarning:` ***Corde***

Chaîne d'avertissement qui correspond à la chaîne d'avertissement sur le fil pour le canal PCF (Programmable Command Formats).

`putAppName:` ***Corde***

Le nom de l'application associé au message MQPUT.

`queue:` ***Corde***

Le nom de la file d'attente locale. La valeur est `null` s'il n'y a pas `MQOPEN`, `MQOPEN_REPLY`, `MQSP1(Open)`, ou `MQSP1_REPLY` message.

`queueMgr:` ***Corde***

Le gestionnaire de files d'attente local. La valeur est `null` s'il n'y a pas `INITIAL_DATA` message au début de la connexion.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `IBMMQ.commitRecord()` sur l'un ou l'autre `IBMMQ_REQUEST` ou `IBMMQ_RESPONSE` événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

DEMANDE IBMMQ	IBMMQ_RESPONSE
canal	canal
Le client est externe	Le client est externe
Client Zerownd	Client Zerownd
ID de corrélation	ID de corrélation
ID MSG	erreur
méthode	ID MSG
Format MSG	méthode
Taille du MSG	Format MSG
file d'attente	Taille du MSG
File d'attente Mgr	file d'attente
Le récepteur est externe	File d'attente Mgr

DEMANDE IBMMQ	IBMMQ_RESPONSE
ReqBytes	Le récepteur est externe
2 octets REQL	File d'attente résolue
ReqPkts	File d'attente résolue Mgr
ReqRTO	Durée de l'aller-retour
File d'attente résolue	Octets RSP
File d'attente résolue Mgr	RSPL 2 octets
L'expéditeur est externe	RSPPKTS
Le serveur est externe	RSP vers
Serveur Zerownd	L'expéditeur est externe
	Le serveur est externe
	Serveur Zerownd
	avertissement

`reqBytes:` ***Numéro***

Le nombre d'octets de demande au niveau de l'application.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`resolvedQueue:` ***Corde***

Le nom de file d'attente résolu provenant de `MQGET_REPLY`, `MQPUT_REPLY`, ou `MQPUT1_REPLY` messages. Si la file d'attente est distante, la valeur est différente de la valeur renvoyée par `IBMMQ.queue`.

`resolvedQueueMgr:` ***Corde***

Le gestionnaire de files d'attente résolu de `MQGET_REPLY`, `MQPUT_REPLY`, ou `MQPUT1_REPLY`. Si la file d'attente est distante, la valeur est différente de la valeur renvoyée par `IBMMQ.queueMgr`.

`rfh:` ***Tableau de chaînes***

Tableau de chaînes situé dans l'en-tête des règles facultatives et de mise en forme (RFH). S' il n'y a pas d'en-tête RFH ou si l'en-tête est vide, le tableau est vide.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`rspBytes:` ***Numéro***

Le nombre d'octets de réponse au niveau de l'application.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse.

`rspPkts:` ***Numéro***

Le nombre de paquets de demandes.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`totalMsgLength:` ***Numéro***

Longueur totale du message, exprimée en octets.

`warning:` ***Corde***

La chaîne d'avertissement qui correspond à la chaîne d'avertissement sur le fil.

Exemples de déclencheurs

- Exemple : collecte des métriques IBMMQ

ICA

Le ICA la classe vous permet de stocker des métriques et d'accéder à des propriétés sur `ICA_OPEN`, `ICA_AUTH`, `ICA_TICK`, et `ICA_CLOSE` événements.

Évènements

`ICA_AUTH`

S'exécute lorsque l'authentification ICA est terminée.

`ICA_CLOSE`

S'exécute lorsque la session de l'ICA est fermée.

`ICA_OPEN`

S'exécute immédiatement après le chargement initial de l'application ICA.

`ICA_TICK`

S'exécute périodiquement pendant que l'utilisateur interagit avec l'application ICA.

Après le `ICA_OPEN` l'événement s'est déroulé au moins une fois, le `ICA_TICK` l'événement est exécuté chaque fois que la latence est signalée et renvoyée par `clientLatency` ou `networkLatency` propriétés décrites ci-dessous.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur `ICA_OPEN`, `ICA_TICK`, ou `ICA_CLOSE` événement. Enregistrez les validations le `ICA_AUTH` les événements ne sont pas pris en charge.

L'événement détermine quelles propriétés sont validées pour l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`application:` ***Corde***

Le nom de l'application en cours de lancement.

`authDomain:` ***Corde***

Le domaine d'authentification Windows auquel appartient l'utilisateur.

`channels:` ***Array***

Tableau d'objets contenant des informations sur les canaux virtuels observés depuis la dernière `ICA_TICK` événement.

Accès uniquement sur `ICA_TICK`événements ; sinon, une erreur se produira.

Chaque objet contient les propriétés suivantes :

`name:` ***Corde***

Le nom du canal virtuel.

`description:` ***Corde***

Description conviviale du nom de la chaîne.

`clientBytes:` ***Numéro***

Le nombre d'octets envoyés par client pour cette chaîne.

`serverBytes:` ***Numéro***

Le nombre d'octets envoyés par le serveur pour le canal.

`clientMachine:` ***Corde***

Le nom du client machine. Le nom est affiché par le client ICA et est généralement le nom d'hôte de la machine cliente.

`clientBytes:` ***Numéro***

Sur un `ICA_CLOSE` événement, le nombre incrémentiel d'octets client au niveau de l'application observés depuis la dernière `ICA_TICK` événement. Ne spécifie pas le nombre total d'octets pour la session.

Accès uniquement sur `ICA_CLOSE` ou `ICA_TICK` événements ; sinon, une erreur se produira.

`clientCGPMsgCount:` ***Numéro***

Le nombre de messages CGP du client depuis le dernier `ICA_TICK` événement.

Accès uniquement sur `ICA_TICK` événements ; sinon, une erreur se produira.

`clientLatency:` ***Numéro***

La latence du client, exprimé en millisecondes, tel que rapporté par la balise de gestion de l'expérience utilisateur final (EUEM).

La latence du client est signalée lorsqu'un paquet du client sur le canal EUEM rapporte le résultat d'une seule mesure aller-retour ICA.

Accès uniquement sur `ICA_TICK` événements ; dans le cas contraire, une erreur se produira.

`clientL2Bytes:` ***Numéro***

Sur un `ICA_CLOSE` événement, le nombre incrémentiel de L2 octets du client observés depuis la dernière `ICA_TICK` événement. Ne spécifie pas le nombre total d'octets pour la session.

Accès uniquement sur `ICA_CLOSE` ou `ICA_TICK` événements ; sinon, une erreur se produira.

`clientMsgCount:` ***Numéro***

Le nombre de messages des clients depuis le dernier `ICA_TICK` événement.

Accès uniquement sur `ICA_TICK` événements ; sinon, une erreur se produira.

`clientPkts:` ***Numéro***

Sur un `ICA_CLOSE` événement, le nombre incrémentiel de paquets clients observés depuis le dernier `ICA_TICK` événement. Ne spécifie pas le nombre total de paquets pour la session.

Accès uniquement sur `ICA_CLOSE` ou `ICA_TICK` événements ; sinon, une erreur se produira.

`clientRTO:` ***Numéro***

Sur un `ICA_CLOSE` événement, le nombre incrémentiel de clients délais de retransmission (RTO) observés depuis la dernière `ICA_TICK` événement. Ne spécifie pas le nombre total de RTO pour la session.

Accès uniquement sur ICA_CLOSE ou ICA_TICK événements ; dans le cas contraire, une erreur se produira.

clientZeroWnd: ***Numéro***

Nombre de fenêtres nulles envoyées par le client.

Accès uniquement sur ICA_CLOSE ou ICA_TICK événements ; sinon, une erreur se produira.

clientType: ***Corde***

Type de client ICA, qui est l'agent utilisateur équivalent à ICA.

clipboardData: ***Tampon***

UNE Tampon objet contenant des données brutes provenant du transfert du presse-papiers.

La valeur est null si le ICA_TICK l'événement n'a pas résulté d'un transfert de données dans le presse-papiers, ou si le canal spécifié par le tickChannel la propriété n'est pas un canal du presse-papiers.

Le nombre maximum d'octets dans la mémoire tampon est spécifié par Octets du presse-papiers à mettre en mémoire tampon champ lorsque le déclencheur a été configuré via le système ExtraHop. La taille d'objet maximale par défaut est de 1 024 octets. Pour plus d'informations, consultez le Options de déclencheur avancées.

Pour déterminer le sens du transfert des données du presse-papiers, accédez à cette propriété via Flow.sender, Flow.receiver, Flow.client, ou Flow.server.

Accès uniquement sur ICA_TICK événements ; dans le cas contraire, une erreur se produira.

clipboardDataType: ***Corde***

Type de données transférées dans le presse-papiers. Les types de presse-papiers suivants sont pris en charge :

- TEXT
- BITMAP
- METAFILEPICT
- SYMLINK
- DIF
- TIFF
- OEMTEXT
- DIB
- PALLETTE
- PENDATA
- RIFF
- WAVE
- UNICODETEXT
- EHNMETAFILE
- OWNERDISPLAY
- DSPTEXT
- DSPBITMAP
- DSPMETAFILEPICT
- DSPENHMETAFILE

La valeur est null si le ICA_TICK l'événement n'a pas résulté d'un transfert de données dans le presse-papiers, ou si le canal spécifié par le tickChannel la propriété n'est pas un canal du presse-papiers.

Accès uniquement sur ICA_TICK événements ; dans le cas contraire, une erreur se produira.

frameCutDuration: ***Numéro***

La durée de coupure de trame, telle que signalée par la balise EUEM.

Accès uniquement sur `ICA_TICK` événements ; dans le cas contraire, une erreur se produira.

`frameSendDuration:` ***Numéro***

La durée d'envoi de la trame, telle que signalée par la balise EUEM.

Accès uniquement sur `ICA_TICK` événements ; dans le cas contraire, une erreur se produira.

`host:` ***Corde***

Le nom d'hôte du serveur Citrix.

`isAborted:` ***Booléen***

La valeur est `true` si l'application ne parvient pas à démarrer correctement.

Accès uniquement sur `ICA_CLOSE` événements ; sinon, une erreur se produira.

`isCleanShutdown:` ***Booléen***

La valeur est `true` si l'application s'arrête correctement.

Accès uniquement sur `ICA_CLOSE` événements ; sinon, une erreur se produira.

`isClientDiskRead:` ***Booléen***

La valeur est `true` si un fichier a été lu depuis le disque client vers le serveur Citrix. La valeur est `null` si la commande n'est pas une opération de fichier, ou si le canal spécifié par `tickChannel` la propriété n'est pas un canal de fichier.

Accès uniquement sur `ICA_TICK` événements ; sinon, une erreur se produira.

`isClientDiskWrite:` ***Booléen***

La valeur est `true` si un fichier a été écrit depuis le serveur Citrix sur le disque client. La valeur est `null` si la commande n'est pas une opération de fichier, ou si le canal spécifié par `tickChannel` la propriété n'est pas un canal de fichier.

Accès uniquement sur `ICA_TICK` événements ; sinon, une erreur se produira.

`isEncrypted:` ***Booléen***

La valeur est `true` si l'application est chiffrée avec le chiffrement RC5 .

`isSharedSession:` ***Booléen***

La valeur est `true` si l'application est lancée via une connexion existante.

`launchParams:` ***Corde***

La chaîne qui représente les paramètres.

`loadTime:` ***Numéro***

Le temps de chargement de l'application donnée, exprimé en millisecondes.

Note: Le temps de chargement est enregistré uniquement pour le chargement initial de l'application. Le système ExtraHop ne mesure pas le temps de chargement des applications lancées au cours de sessions existantes, mais indique le temps de chargement initial lors des chargements d'applications suivants. Choisissez `ICA.isSharedSession` pour faire la distinction entre les charges d'application initiales et suivantes.

`loginTime:` ***Numéro***

Temps de connexion de l'utilisateur, exprimé en millisecondes.

Accès uniquement sur `ICA_OPEN`, `ICA_CLOSE`, ou `ICA_TICK` événements ; dans le cas contraire, une erreur se produira.

Note: L'heure de connexion est enregistrée uniquement pour le chargement initial de l'application. Le système ExtraHop ne mesure pas le temps de connexion pour les applications lancées au cours de sessions existantes, mais indique l' heure de connexion initiale lors des chargements d'applications suivants. Choisissez `ICA.isSharedSession` pour faire la distinction entre les charges d'application initiales et suivantes.

`networkLatency:` ***Numéro***

La latence actuelle annoncée par le client, exprimé en millisecondes.

La latence du réseau est signalée lorsqu'un paquet ICA spécifique provenant du client contient des informations de latence.

Accès uniquement sur `ICA_TICK` événements ; dans le cas contraire, une erreur se produira.

`payload:` ***Tampon***

Le Tampon objet contenant les octets de charge utile bruts du fichier lu ou écrit lors de l'événement.

La mémoire tampon contient *N* premiers octets de la charge utile, où *N* est le nombre d'octets de charge utile spécifié par Octets vers la mémoire tampon champ lorsque le déclencheur a été configuré via l'interface utilisateur Web ExtraHop. Le nombre d' octets par défaut est de 2 048. Pour plus d'informations, voir Options de déclencheur avancées.

La valeur est `null` si le canal spécifié par `tickChannel` la propriété n'est pas un canal de fichier.

Accès uniquement sur `ICA_TICK` événements ; sinon, une erreur se produira.

`printerName:` ***Corde***

Le nom du pilote dquelconque de l'imprimante.

Accès uniquement sur `ICA_TICK` événements ; sinon, une erreur se produira.

`program:` ***Corde***

Nom du programme ou de l'application en cours de lancement.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `ICA.commitRecord()` sur l'un ou l'autre `ICA_OPEN`, `ICA_TICK`, ou `ICA_CLOSE` événement.

L'événement pour lequel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

ICA_CLOSE	ICA_OPEN	ICA_TICK
authDomain	authDomain	authDomain
clientBytes	clientIsExternal	clientIsExternal
clientIsExternal	clientMachine	clientBytes
clientL2Bytes	clientType	clientCGPMsgCount
clientMachine	clientZeroWnd	clientL2Bytes
clientPkts	host	clientLatency
clientRTO	isEncrypted	clientMachine
clientType	isSharedSession	clientMsgCount
clientZeroWnd	launchParams	clientPkts
host	loadTime	clientRTO
isAborted	loginTime	clientType
isCleanShutdown	program	clientZeroWnd
isEncypted	receiverIsExternal	frameCutDuration
isSharedSession	senderIsExternal	frameSendDuration
launchParams	serverIsExternal	host

ICA_CLOSE	ICA_OPEN	ICA_TICK
loadTime	serverZeroWnd	isClientDiskRead
loginTime	user	isClientDiskWrite
program		isEncrypted
receiverIsExternal		isSharedSession
roundTripTime		launchParams
senderIsExternal		loadTime
serverBytes		loginTime
serverIsExternal		networkLatency
serverL2Bytes		program
serverPkts		receiverIsExternal
serverRTO		resource
serverZeroWnd		roundTripTime
user		senderIsExternal
		serverBytes
		serverCGPMsgCount
		serverIsExternal
		serverL2Bytes
		serverMsgCount
		serverPkts
		serverRTO
		serverZeroWnd
		tickChannel
		user

Accédez à l'objet d'enregistrement uniquement sur ICA_OPEN, ICA_CLOSE, et ICA_TICK événements ; sinon, une erreur se produira.

`resource:` ***Corde***

Le chemin du fichier qui a été lu ou écrit lors de l'événement, s'il est connu. La valeur est `null` si le canal spécifié par `tickChannel` la propriété n'est pas un canal de fichier.

Accès uniquement sur ICA_TICK événements ; sinon, une erreur se produira.

`resourceOffset:` ***Numéro***

Le décalage du fichier lu ou écrit lors de l'événement, s'il est connu. La valeur est `null` si le canal spécifié par `tickChannel` la propriété n'est pas un canal de fichier.

Accès uniquement sur ICA_TICK événements ; sinon, une erreur se produira.

`roundTripTime:` ***Numéro***

Le temps d'aller-retour médian (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur ICA_CLOSE ou ICA_TICK événements ; sinon, une erreur se produira.

`serverBytes:` ***Numéro***

Sur un `ICA_CLOSE` événement, le nombre incrémentiel d'octets de serveur au niveau de l'application observés depuis la dernière `ICA_TICK` événement. Ne spécifie pas le nombre total d'octets pour la session.

Accès uniquement sur `ICA_CLOSE` ou `ICA_TICK` événements ; sinon, une erreur se produira.

`serverCGPMsgCount:` ***Numéro***

Le nombre de messages du serveur CGP depuis le dernier `ICA_TICK` événement.

Accès uniquement sur `ICA_TICK` événements ; sinon, une erreur se produira.

`serverL2Bytes:` ***Numéro***

Sur un `ICA_CLOSE` événement, le nombre incrémentiel de L2 octets de serveur observés depuis la dernière `ICA_TICK` événement. Ne spécifie pas le nombre total d'octets pour la session.

Accès uniquement sur `ICA_CLOSE` ou `ICA_TICK` événements ; sinon, une erreur se produira.

`serverMsgCount:` ***Numéro***

Le nombre de messages du serveur depuis le dernier `ICA_TICK` événement.

Accès uniquement sur `ICA_TICK` événements ; sinon, une erreur se produira.

`serverPkts:` ***Numéro***

Sur un `ICA_CLOSE` événement, le nombre incrémentiel de paquets de serveur observés depuis le dernier `ICA_TICK` événement. Ne spécifie pas le nombre total de paquets pour la session.

Accès uniquement sur `ICA_CLOSE` ou `ICA_TICK` événements ; sinon, une erreur se produira.

`serverRTO:` ***Numéro***

Sur un `ICA_CLOSE` événement, le nombre incrémentiel de serveurs délais de retransmission (RTO) observés depuis la dernière `ICA_TICK` événement. Ne spécifie pas le nombre total de RTO pour la session.

Accès uniquement sur `ICA_CLOSE` ou `ICA_TICK` événements ; dans le cas contraire, une erreur se produira.

`serverZeroWnd:` ***Numéro***

Nombre de fenêtres nulles envoyées par le serveur.

Accès uniquement sur `ICA_CLOSE` ou `ICA_TICK` événements ; sinon, une erreur se produira.

`tickChannel:` ***Corde***

Le nom du canal virtuel qui a généré le courant `ICA_TICK` événement. Les chaînes suivantes sont prises en charge :

- **CTXCLI:** Presse-papiers
- **CTXCDM:** Dossier
- **CTXUE:** Surveillance de l'expérience de l'utilisateur final

Accès uniquement sur `ICA_TICK` événements ; sinon, une erreur se produira.

`user:` ***Corde***

Le nom de l'utilisateur, s'il est disponible.

ICMP

Le ICMP la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `ICMP_MESSAGE` événements.

Évènements

`ICMP_MESSAGE`

S'exécute sur tous les messages ICMP traités par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `ICMP_MESSAGE` événement.

Pour consulter les propriétés par défaut attribuées à l'objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`gwAddr:` ***Adresse IP***

Pour un message de redirection, renvoie l'adresse de la passerelle vers laquelle le trafic du réseau spécifié dans le champ réseau de destination Internet des données du datagramme d'origine doit être envoyé. Renvoie null pour tous les autres messages.

Message	Type d'ICMPv4	Type d'ICMPv6
`Redirect Message`	`5`	`n/a`

`hopLimit:` ***Numéro***

Durée de vie du paquet ICMP ou nombre de sauts.

`isError:` ***Booléen***

La valeur est `true` pour les types de messages figurant dans le tableau suivant.

Un message	Type d'ICMPv4	Type d'ICMPv6
`Destination Unreachable`	`3`	`1`
`Redirect`	`5`	`n/a`
`Source Quench`	`4`	`n/a`
`Time Exceeded`	`11`	`3`
`Parameter Problem`	`12`	`4`
`Packet Too Big`	`n/a`	`2`

`isQuery:` ***Booléen***

La valeur est `true` pour les types de messages figurant dans le tableau suivant.

Un message	Type d'ICMPv4	Type d'ICMPv6
`Echo Request`	`8`	`128`
`Information Request`	`15`	`n/a`
`Timestamp request`	`13`	`n/a`
`Address Mask Request`	`17`	`n/a`
`Router Discovery`	`10`	`151`

Un message	Type d'ICMPv4	Type d'ICMPv6
Multicast Listener Query	n/a	130
Router Solicitation (NDP)	n/a	133
Neighbor Solicitation	n/a	135
ICMP Node Information Query	n/a	139
Inverse Neighbor Discovery Solicitation	n/a	141
Home Agent Address Discovery Solicitation	n/a	144
Mobile Prefix Solicitation	n/a	146
Certification Path Solicitation	n/a	148

`isReply:` ***Booléen***

La valeur est `true` pour les types de messages figurant dans le tableau suivant.

Un message	Type d'ICMPv4	Type d'ICMPv6
Echo Reply	0	129
Information Reply	16	n/a
Timestamp Reply	14	n/a
Address Mask Reply	18	n/a
Multicast Listener Done	n/a	132
Multicast Listener Report	n/a	131
Router Advertisement (NDP)	n/a	134
Neighbor Advertisement	n/a	136
ICMP Node Information Response	n/a	140
Inverse Neighbor Discovery Advertisement	n/a	142
Home Agent Address Discovery Reply Message	n/a	145
Mobile Prefix Advertisement	n/a	147
Certification Path Advertisement	n/a	149

`msg:` ***Tampon***

Un objet tampon contenant jusqu'à `message_length_max` octets du message ICMP . Le `message_length_max` l'option est configurée dans le profil ICMP de la configuration en cours.

L'exemple de configuration d'exécution suivant modifie l'ICMP `message_length_max` de sa valeur par défaut de 4096 octets à 1234 octets :

```
"capture": {
    "app_proto": {
        "ICMP": {
            "message_length_max": 1234
        }
```

```
    }
}
```

Conseil Vous pouvez convertir l'objet tampon en chaîne à l'aide de la méthode String.fromCharCode. Pour afficher la chaîne dans le journal d'exécution, exécutez la méthode JSON.stringify, comme indiqué dans l'exemple de code suivant :

```
const icmp_msg = String.fromCharCode.apply(String,
 ICMP.msg);
debug('ICMP message text: ' + JSON.stringify(icmp_msg,
 null, 4));
```

Vous pouvez également rechercher les chaînes de messages ICMP à l'aide des inclusions et des méthodes de test, comme illustré dans l'exemple de code suivant :

```
const substring_search = 'search term';
const regex_search = '^search term$';
const icmp_msg = String.fromCharCode.apply(String,
 ICMP.msg);

if (icmp_msg.includes(substring_search){
    debug('ICMP message includes substring');
}
if (regex_search.test(icmp_msg)){
    debug('ICMP message matches regex');
}
```

`msgCode:` ***Numéro***

Le code du message ICMP.

`msgId:` ***Numéro***

Identifiant de message ICMP pour les messages Echo Request, Echo Reply, Timestamp Request, Timestamp Reply, Information Request et Information Reply. La valeur est `null` pour tous les autres types de messages.

Le tableau suivant affiche les identifiants de type pour les messages ICMP :

Un message	Type d'ICMPv4	Type d'ICMPv6
Echo Request	8	128
Echo Reply	0	129
Timestamp Request	13	n/a
Timestamp Reply	14	n/a
Information Request	15	n/a
Information Reply	16	n/a

`msgLength:` ***Numéro***

Longueur du message ICMP, exprimée en octets.

`msgText:` ***Corde***

Le texte descriptif du message (par exemple, demande d'écho ou port inaccessible).

`msgType:` ***Numéro***

Le type de message ICMP.

Le tableau suivant indique les types de messages ICMPv4 disponibles :

Type	Un message
0	Echo Reply
1 and 2	Unassigned
3	Destination Unreachable
4	Source Quench
5	Redirect Message
6	Alternate Host Address (deprecated)
7	Unassigned
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded
12	Parameter Problem: Bad IP header
13	Timestamp
14	Timestamp Reply
15	Information Request (deprecated)
16	Information Reply (deprecated)
17	Address Mask Request (deprecated)
18	Address Mask Reply (deprecated)
19	Reserved
20-29	Reserved
30	Traceroute (deprecated)
31	Datagram Conversion Error (deprecated)
32	Mobile Host Redirect (deprecated)
33	Where Are You (deprecated)
34	Here I Am (deprecated)
35	Mobile Registration Request (deprecated)
36	Mobile Registration Reply (deprecated)
37	Domain Name Request (deprecated)
38	Domain Name Reply (deprecated)
39	Simple Key-Management for Internet Protocol (deprecated)
40	Photuris (deprecated)
41	ICMP experimental
42	Extended Echo Request
43	Extended Echo Reply

Type	Un message
44-255	Unassigned

Le tableau suivant indique les types de messages ICMPv6 disponibles :

Type	Un message
1	Destination Unreachable
2	Packet Too Big
3	Time Exceeded
4	Parameter Problem
100	Private Experimentation
101	Private Experimentation
127	Reserved for expansion of ICMPv6 error messages
128	Echo Request
129	Echo Reply
130	Multicast Listener Query
131	Multicast Listener Report
132	Multicast Listener Done
133	Router Solicitation
134	Router Advertisement
135	Neighbor Solicitation
136	Neighbor Advertisement
137	Redirect Message
138	Router Renumbering
139	ICMP Node Information Query
140	ICMP Node Information Response
141	Inverse Neighbor Discovery Solicitation Message
142	Inverse Neighbor Discovery Advertisement Message
143	Multicast Listener Discovery (MLDv2) reports
144	Home Agent Address Discovery Request Message
145	Home Agent Address Discovery Reply Message
146	Mobile Prefix Solicitation
147	Mobile Prefix Advertisement
148	Certification Path Solicitation
149	Certification Path Advertisement
150	ICMP messages utilized by experimental mobility protocols such as Seamoby

Type	Un message
151	Multicast Router Advertisement
152	Multicast Router Solicitation
153	Multicast Router Termination
155	RPL Control Message
156	ILNPv6 Locator Update Message
157	Duplicate Address Request
158	Duplicate Address Confirmation
159	MPL Control Message
160	Extended Echo Request - No Error
161	Extended Echo Reply
200	Private Experimentation
201	Private Experimentation
255	Reserved for expansion of ICMPv6 informational messages

`nextHopMTU:` ***Numéro***

Un ICMPv4 `Destination inaccessible` ou un ICMPv6 `Le paquet est trop gros` message, l'unité de transmission maximale du lien du saut suivant. La valeur est `null` pour tous les autres messages.

Un message	Type d'ICMPv4	Type d'ICMPv6
Destination Unreachable	3	n/a
Packet Too Big	n/a	2

`original:` ***Objet***

Objet contenant les éléments suivants du datagramme IP à l'origine de l'envoi du message ICMP :

`ipproto:` ***Corde***

Le protocole IP du datagramme, tel que TCP, UDP, ICMP ou ICMPv6.

`ipver:` ***Corde***

Version IP du datagramme, telle que IPv4 ou IPv6.

`srcAddr:` ***Adresse IP***

Le IPAddress de l'expéditeur du datagramme.

`srcPort:` ***Numéro***

Numéro de port de l'expéditeur du datagramme.

`dstAddr:` ***Adresse IP***

Le IPAddress du récepteur de datagrammes.

`dstPort:` ***Numéro***

Numéro de port du récepteur de datagrammes.

La valeur est `null` si l'en-tête Internet et 64 bits du datagramme des données d'origine ne sont pas présents dans le message ou si le protocole IP n'est pas TCP ou UDP.

Accès uniquement sur `ICMP_MESSAGE` événements ; dans le cas contraire, une erreur se produira.

`pointer:` ***Numéro***

Pour un message de problème de paramètre, octet de l'en-tête du datagramme d'origine où l' erreur a été détectée. La valeur est `null` pour tous les autres messages.

Un message	Type d'ICMPv4	Type d'ICMPv6
`Parameter Problem`	`12`	`4`

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `ICMP.commitRecord()` sur un `ICMP_MESSAGE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `gwAddr`
- `hopLimit`
- `msgCode`
- `msgId`
- `msgLength`
- `msgText`
- `msgType`
- `nextHopMTU`
- `pointer`
- `receiverIsExternal`
- `senderIsExternal`
- `serverIsExternal`
- `seqNum`
- `version`

`seqNum:` ***Numéro***

Numéro de séquence ICMP pour les messages de demande d'écho, de réponse d'écho, de demande d'horodatage, de réponse d'horodatage, de demande d'information et de réponse d'information. La valeur est `null` pour tous les autres messages.

`version:` ***Numéro***

Version du type de message ICMP, qui peut être ICMPv4 ou ICMPv6.

`Kerberos`

Le Kerberos la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `KERBEROS_REQUEST` et `KERBEROS_RESPONSE` événements.

Évènements

`KERBEROS_REQUEST`

Fonctionne sur tous les types de messages Kerberos AS-REQ et TGS-REQ traités par l'équipement.

`KERBEROS_RESPONSE`

Fonctionne sur tous les types de messages Kerberos AS-REP et TGS-REP traités par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `KERBEROS_REQUEST` ou `KERBEROS_RESPONSE` événement.

L' événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si le `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`addresses:` ***Tableau d'objets***

Les adresses à partir desquelles le billet demandé est valide.

Accès uniquement sur KERBEROS_REQUEST événements ; dans le cas contraire, une erreur se produira.

`apOptions:` ***Objet***

Un objet contenant des valeurs booléennes pour chaque indicateur d'option dans les messages AP_REQ.

Accès uniquement sur KERBEROS_REQUEST événements ; dans le cas contraire, une erreur se produira.

`clientPrincipalName:` ***Corde***

Le nom principal du client.

`cNames:` ***Tableau de chaînes***

Les parties du nom de l'identifiant principal.

`cNameType:` ***Corde***

Type du champ CNames.

`cRealm:` ***Corde***

Le domaine du client.

`eData:` ***Tampon***

Informations supplémentaires sur l'erreur renvoyée dans la réponse.

Accès uniquement sur KERBEROS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`error:` ***Corde***

L'erreur est retournée.

Accès uniquement sur KERBEROS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`from:` ***Corde***

Dans les types de message AS_REQ et TGS_REQ, l'heure à laquelle le ticket demandé doit être postdaté.

Accès uniquement sur KERBEROS_REQUEST événements ; dans le cas contraire, une erreur se produira.

`isAccountPrivileged:` ***Booléen***

La valeur est vraie si le compte spécifié dans le `clientPrincipalName` la propriété est privilégiée.

`kdcOptions:` ***Objet***

Un objet contenant des valeurs booléennes pour chaque indicateur d'option dans les messages AS_REQ et TGS_REQ.

Accès uniquement sur KERBEROS_REQUEST événements ; dans le cas contraire, une erreur se produira.

`msgType:` ***Corde***

Le type de message. Les valeurs possibles sont les suivantes :

- AP_REP
- AP_REQ
- AS_REP
- AS_REQAUTHENTICATOR
- ENC_AS_REP_PART
- ENC_KRB_CRED_PART
- ENC_KRB_PRIV_PART
- ENC_P_REP_PART
- ENC_TGS_REP_PART
- ENC_TICKET_PART
- KRB_CRED
- KRB_ERROR
- KRB_PRIV
- KRB_SAFE
- TGS_REP
- TGS_REQ
- TICKET

`paData:` ***Tableau d'objets***

Les données de pré-authentification.

`processingTime:` ***Numéro***

Le temps de traitement, exprimé en millisecondes.

Accès uniquement sur KERBEROS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`realm:` ***Corde***

Le domaine du serveur. Dans un message de type AS_REQ, il s'agit du domaine client.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `Kerberos.commitRecord()` sur l'un ou l'autre KERBEROS_REQUEST ou KERBEROS_RESPONSE événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

KERBEROS_REQUEST	KERBEROS_RESPONSE
clientIsExternal	clientIsExternal
cNames	cNames
cNameType	cNameType
cRealm	cRealm
clientZeroWnd	clientZeroWnd
encryptedTicketLength	encryptedTicketLength
eType	error
from	msgType
isAccountPrivileged	isAccountPrivileged

KERBEROS_REQUEST	KERBEROS_RESPONSE
msgType	processingTime
realm	realm
receiverIsExternal	receiverIsExternal
reqBytes	roundTripTime
reqL2Bytes	rspBytes
reqPkts	rspL2Bytes
reqRTO	rspPkts
senderIsExternal	rspRTO
serverZeroWnd	senderIsExternal
sNames	serverIsExternal
sNameType	sNames
ticketETypeName	sNameType
till	ticketETypeName
	serverZeroWnd

`reqETypes:` ***Tableau de nombres***

Tableau de nombres correspondant aux méthodes de chiffrement préférées.

Méthode de chiffrement	Numéro
ntlm-hash	-150
aes256-cts-hmac-sha1-96-plain	-149
aes128-cts-hmac-sha1-96-plain	-148
rc4-plain-exp	-141
rc4-plain	-140
rc4-plain-old-exp	-136
rc4-hmac-old-exp	-135
rc4-plain-old	-134
rcr-hmac-old	-133
des-plain	-132
rc4-sha	-131
rc4-lm	-130
rc4-plain2	-129
rc4-md4	-128
null	0
des-cbc-crc	1
des-cbc-md4	2

Méthode de chiffrement	Numéro
des-cbc-md5	3
des3-cbc-md5	5
des3-cbc-sha1	7
dsaWithSHA1-CmsOID	9
md5WithRSAEncryption-CmsOID	10
sha1WithRSAEncryption-CmsOID	11
rc2CBC-EnvOID	12
rsaEncryption-EnvOID	13
rsaES-OAEP-ENV-OID	14
des-ede3-cbc-Env-OID	15
des3-cbc-sha1-kd	16
aes128-cts-hmac-sha1-96	17
aes256-cts-hmac-sha1-96	18
aes128-cts-hmac-sha256-128	19
aes256-cts-hmac-sha384-192	20
rc4-hmac	23
rc4-hmac-exp	24
camellia128-cts-cmac	25
camellia256-cts-cmac	26
subkey-keymaterial	65

`reqETypeNames:` ***Tableau de chaînes***

Tableau des méthodes de chiffrement préférées.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`serverPrincipalName:` ***Corde***

Le nom principal du serveur (SPN).

`sNames:` ***Tableau de chaînes***

Les parties du nom de l'identifiant principal du serveur.

`sNameType:` ***Corde***

Type du champ SNames.

`ticket:` ***Objet***

Un ticket nouvellement généré dans un message AP_REP ou un ticket pour authentifier le client auprès du serveur dans un message AP_REQ.

`till:` ***Corde***

Date d'expiration demandée par le client dans une demande de ticket.

Accès uniquement sur KERBEROS_REQUEST événements ; dans le cas contraire, une erreur se produira.

LDAP

Le LDAP la classe vous permet de stocker des métriques et d'accéder à des propriétés sur LDAP_REQUEST et LDAP_RESPONSE événements.

Évènements

LDAP_REQUEST

S'exécute sur chaque demande LDAP traitée par l'équipement.

LDAP_RESPONSE

S'exécute sur chaque réponse LDAP traitée par l'équipement.

Méthodes

`commitRecord()`: ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur LDAP_REQUEST ou LDAP_RESPONSE événement.

L'événement détermine quelles propriétés sont validées pour l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`bindDN`: ***Corde***

Le DN de liaison de la demande LDAP.

Accès uniquement sur LDAP_REQUEST événements ; dans le cas contraire, une erreur se produira.

`controls`: ***Tableau d'objets***

Tableau d'objets contenant les contrôles LDAP de la demande LDAP. Chaque objet contient les propriétés suivantes :

`controlType`: ***Corde***

L'OID du contrôle LDAP.

`criticality`: ***Booléen***

Indique si le contrôle est requis. Si `criticality` est réglé sur `true`, le serveur doit traiter le contrôle ou échouer l' opération.

`controlValue`: ***Tampon***

La valeur de contrôle facultative, qui spécifie des informations supplémentaires sur la manière dont le contrôle doit être traité.

Accès uniquement sur LDAP_REQUEST événements ; dans le cas contraire, une erreur se produira.

`dn`: ***Corde***

Le nom distinctif (DN) LDAP. Si aucun DN n'est défini, <ROOT> sera retourné à la place.

`encryptionProtocol`: ***Corde***

Le protocole avec lequel la transaction est cryptée.

`error`: ***Corde***

La chaîne d'erreur LDAP courte telle que définie dans le protocole (par exemple, NoSuchObject).

Accès uniquement sur LDAP_RESPONSE événements ; sinon, une erreur se produira.

Code de résultat	Chaîne de résultats
1	operationsError
2	protocolError
3	timeLimitExceeded
4	sizeLimitExceeded
7	authMethodNotSupported
8	strongerAuthRequired
11	adminLimitExceeded
12	unavailableCriticalExtension
13	confidentialityRequired
16	noSuchAttribute
17	undefinedAttributeType
18	inappropriateMatching
19	constraintViolation
20	attributeOrValueExists
21	invalidAttributeSyntax
32	NoSuchObject
33	aliasProblem
34	invalidDNSSyntax
36	aliasDeferencingProblem
48	inappropriateAuthentication
49	invalidCredentials
50	insufficientAccessRights
51	busy
52	unavailable
53	unwillingToPerform
54	loopDetect
64	namingViolation
65	objectClassViolation
66	notAllowedOnNonLeaf
67	notAllowedOnRDN
68	entryAlreadyExists
69	objectClassModsProhibited
71	affectsMultipleDSAs

Code de résultat	Chaîne de résultats
80	other

`errorDetail:` ***Corde***

Le détail de l'erreur LDAP, s'il est disponible pour le type d'erreur. Par exemple, « ProtocolError : version historique du protocole demandée, utilisez plutôt LDAPv3 ».

Accès uniquement sur LDAP_RESPONSE événements ; sinon, une erreur se produira.

`isEncrypted:` ***Booléen***

La valeur est vraie si la transaction est cryptée avec le protocole TLS.

`isDecrypted:` ***Booléen***

La valeur est vraie si le système ExtraHop a déchiffré et analysé la transaction de manière sécurisée. L'analyse du trafic déchiffré peut révéler des menaces avancées qui se cachent dans le trafic chiffré.

`isPasswordEmpty:` ***Booléen***

La valeur est vraie si la demande ne spécifie pas de mot de passe pour l'authentification.

Accès uniquement sur LDAP_REQUEST événements ; dans le cas contraire, une erreur se produira.

`isSigned:` ***Booléen***

La valeur est vraie si la transaction LDAP a été signée par la machine source.

`method:` ***Corde***

La méthode LDAP.

`msgSize:` ***Numéro***

Taille du message LDAP, exprimée en octets.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. La valeur est `NaN` sur les réponses mal formées et abandonnées, si le timing n' est pas valide ou si le timing n'est pas disponible. Disponible pour les appareils suivants :

- `BindRequest`
- `SearchRequest`
- `ModifyRequest`
- `AddRequest`
- `DelRequest`
- `ModifyDNRequest`
- `CompareRequest`
- `ExtendedRequest`

S'applique uniquement à LDAP_RESPONSE événements.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `LDAP.commitRecord()` sur l'un ou l'autre LDAP_REQUEST ou LDAP_RESPONSE événement.

L'événement pour lequel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

LDAP_REQUEST	LDAP_RESPONSE
bindDN	clientIsExternal
clientIsExternal	clientZeroWnd
clientZeroWnd	dn

LDAP_REQUEST	LDAP_RESPONSE
dn	error
isSigned	isSigned
method	errorDetail
msgSize	method
receiverIsExternal	msgSize
reqBytes	processingTime
reqL2Bytes	receiverIsExternal
reqPkts	roundTripTime
reqRTO	rspBytes
saslMechanism	rspL2Bytes
searchFilter	rspPkts
searchScope	rspRTO
senderIsExternal	saslMechanism
serverIsExternal	senderIsExternal
serverZeroWnd	serverIsExternal
	serverZeroWnd

`reqBytes:` ***Numéro***
Le nombre de L4 octets de requête, à l'exclusion des en-têtes L4.

`reqL2Bytes:` ***Numéro***
Le nombre de L2 octets de requête, y compris les en-têtes L2.

`reqPkts:` ***Numéro***
Le nombre de paquets de requêtes.

`reqRTO:` ***Numéro***
Le numéro de demande délais de retransmission (RTO).

`reqZeroWnd:` ***Numéro***
Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***
Le temps médian aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`rspBytes:` ***Numéro***
Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

`rspL2Bytes:` ***Numéro***
Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

`rspPkts:` ***Numéro***
Le nombre de paquets de réponse.

`rspRTO:` ***Numéro***
Le nombre de réponses délais de retransmission (RTO).

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`saslMechanism:` ***Corde***

Chaîne qui définit le mécanisme SASL qui identifie et authentifie un utilisateur auprès d'un serveur.

`searchAttributes:` ***Array***

Les attributs à renvoyer à partir des objets qui correspondent aux critères du filtre.

Accès uniquement sur `LDAP_REQUEST` événements ; sinon, une erreur se produira.

`searchFilter:` ***Corde***

Le mécanisme permettant d'autoriser certaines entrées dans la sous-arborescence et d'en exclure d'autres.

Accès uniquement sur `LDAP_REQUEST` événements ; sinon, une erreur se produira.

`searchResults:` ***Tableau d'objets***

Tableau d'objets contenant les résultats de recherche renvoyés dans une réponse LDAP. Chaque objet contient les propriétés suivantes :

`type:` ***Corde***

Type de résultat de recherche.

`values:` ***Tableau de buffers***

Tableau d'objets Buffer contenant les valeurs des résultats de recherche.

Accès uniquement sur `LDAP_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`searchScope:` ***Corde***

Profondeur d'une recherche dans la base de recherche.

Accès uniquement sur `LDAP_REQUEST` événements ; sinon, une erreur se produira.

`LLDP`

Le LLDP la classe vous permet d'accéder aux propriétés sur `LLDP_FRAME` événements.

Évènements

`LLDP_FRAME`

S'exécute sur chaque trame LLDP traitée par l'équipement.

Propriétés

`chassisId:` ***Tampon***

L'ID du châssis, obtenu à partir du champ de données ChassisID, ou valeur de longueur de type (TLV).

`chassisIdSubtype:` ***Numéro***

Le sous-type d'ID de châssis, obtenu à partir du ChassisID TLV.

`destination:` ***Corde***

Adresse MAC de destination. Adresse MAC de destination. Les destinations les plus courantes sont `01-80-C2-00-00-00`, `01-80-C2-00-00-03` et `01-80-C2-00-00-0E`, indiquant les adresses de multidiffusion.

`optTLVs:` ***Array***

Un tableau contenant les TLV facultatifs. Chaque TLV est un objet possédant les propriétés suivantes :

`customSubtype:` ***Numéro***

Sous-type d'un TLV spécifique à une organisation.

`isCustom:` ***Booléen***

Renvoie vrai si l'objet est un TLV spécifique à l'organisation.

`oui:` ***Numéro***

Identifiant unique pour les TLV spécifiques à l'organisation.

`type:` ***Numéro***

Le type de TLV.

`value:` ***Corde***

La valeur du TLV.

`portId:` ***Tampon***

L'ID du port, obtenu à partir du portID TLV.

`portIdSubtype:` ***Numéro***

Le sous-type d'ID de port, obtenu à partir du TLV PortID.

`source:` ***Appareil***

L'équipement qui envoie la trame LLDP.

`ttl:` ***Numéro***

Le temps de vie, exprimé en secondes. Il s'agit de la durée pendant laquelle les informations contenues dans ce cadre sont valides, à compter de leur réception.

`LLMNR`

Le `LLMNR` la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `LLMNR_REQUEST` et `LLMNR_RESPONSE` événements.

Évènements

`LLMNR_REQUEST`

S'exécute sur chaque demande LLMNR traitée par l'équipement.

`LLMNR_RESPONSE`

S'exécute sur chaque réponse LLMNR traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `LLMNR_REQUEST` ou `LLMNR_RESPONSE` événement.

L'événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut attribuées à l' objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`answer:` ***Objet***

Un objet qui correspond à un enregistrement de ressource de réponse.

Accès uniquement sur `LLMNR_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

Les objets contiennent les propriétés suivantes :

`data:` ***Corde | Adresse IP***

La valeur des données dépend du type. La valeur est `null` pour les types d'enregistrement non pris en charge. Les types d'enregistrement pris en charge sont les suivants :

- A
- AAAA
- NS
- PTR
- CNAME
- MX
- SRV
- SOA
- TXT

`name:` ***Corde***

Le nom de l'enregistrement.

`ttl:` ***Numéro***

La valeur du temps de vie.

`type:` ***Corde***

Type d'enregistrement LLMNR.

`error:` ***Corde***

Nom du code d'erreur LLMNR, conformément aux paramètres LLMNR de l'IANA.

Renvoie OTHER pour les codes d'erreur non reconnus par le système ; toutefois, `errorNum` indique la valeur du code numérique.

Accès uniquement sur `LLMNR_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`errorNum:` ***Numéro***

Représentation numérique du code d'erreur LLMNR conformément aux paramètres LLMNR de l'IANA.

Accès uniquement sur `LLMNR_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`opcode:` ***Corde***

Nom du code d'opération LLMNR conformément aux paramètres LLMNR de l'IANA. Les codes suivants sont reconnus par le système ExtraHop :

Code postal	Nom
0	Query
1	IQuery (Inverse Query - Obsolete)
2	Status
3	Unassigned
4	Notify
5	Update
6-15	Unassigned

Renvoie OTHER pour les codes qui ne sont pas reconnus par le système ; toutefois, le `opcodeNum` propriété spécifie la valeur du code numérique.

`opcodeNum:` ***Numéro***

Représentation numérique du code d'opération LLMNR conformément aux paramètres LLMNR de l'IANA.

`qname:` ***Corde***

Le nom d'hôte demandé.

`qtype:` ***Corde***

Nom du type d'enregistrement de demande LLMNR conformément aux paramètres LLMNR de l'IANA.

Renvoie OTHER pour les types qui ne sont pas reconnus par le système ; toutefois, le `qtypeNum` propriété spécifie la valeur du type numérique.

`qtypeNum:` ***Numéro***

Représentation numérique du type d'enregistrement de demande LLMNR conformément aux paramètres LLMNR de l'IANA .

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `LLMNR.commitRecord()` sur l'un ou l'autre `LLMNR_REQUEST` ou `LLMNR_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

LLMNR_REQUEST	LLMNR_RESPONSE
clientIsExternal	answer
opcode	clientIsExternal
qname	error
qtype	opcode
receiverIsExternal	qname
reqBytes	qtype
reqL2Bytes	receiverIsExternal
reqPkts	rspBytes
senderIsExternal	rspL2Bytes
serverIsExternal	rspPkts
	senderIsExternal
	serverIsExternal

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

Accès uniquement sur `LLMNR_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

Accès uniquement sur `LLMNR_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

Accès uniquement sur `LLMNR_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur LLMNR_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris les surcharges liées au protocole, telles que les en-têtes.

Accès uniquement sur LLMNR_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre d'octets de réponse au niveau de l'application.

Accès uniquement sur LLMNR_RESPONSE événements ; dans le cas contraire, une erreur se produira.

Memcache

Le Memcache la classe vous permet de stocker des métriques et d'accéder aux propriétés sur MEMCACHE_REQUEST et MEMCACHE_RESPONSE événements.

Évènements

MEMCACHE_REQUEST

S'exécute sur chaque demande de cache mémoire traitée par l'équipement.

MEMCACHE_RESPONSE

S'exécute sur chaque réponse de cache mémoire traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un MEMCACHE_REQUEST ou MEMCACHE_RESPONSE événement.

L' événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`accessTime:` ***Numéro***

Le temps d'accès, exprimé en millisecondes. Disponible uniquement si la première touche demandée a généré un résultat.

Accès uniquement sur MEMCACHE_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`error:` ***Corde***

Le message d'erreur détaillé enregistré par le système ExtraHop.

Accès uniquement sur MEMCACHE_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`hits:` ***Array***

Un tableau d'objets contenant la clé Memcache et la taille de la clé.

Accès uniquement sur MEMCACHE_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`key:` ***Corde | nul***

La clé Memcache pour laquelle cela a été activé, si elle est disponible.

`size:` ***Numéro***

Taille de la valeur renvoyée pour la clé, exprimée en octets.

`isBinaryProtocol:` ***Booléen***

La valeur est `true` si la demande/réponse correspond à la version binaire du protocole memcache.

`isNoReply:` ***Booléen***

La valeur est `true` si la demande contient le mot clé « noreply » et ne doit donc jamais recevoir de réponse (protocole texte uniquement).

Accès uniquement sur `MEMCACHE_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`isRspImplicit:` ***Booléen***

La valeur est `true` si la réponse était implicite par une réponse ultérieure du serveur (protocole binaire uniquement).

Accès uniquement sur `MEMCACHE_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`method:` ***Corde***

La méthode Memcache telle qu'elle est enregistrée dans la section Metrics du système ExtraHop.

`misses:` ***Array***

Tableau d'objets contenant la clé Memcache.

Accès uniquement sur `MEMCACHE_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`key:` ***Corde | nul***

La clé Memcache pour laquelle cela n'a pas été fait, si elle est disponible.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `Memcache.commitRecord()` sur l'un ou l'autre `MEMCACHE_REQUEST` ou `MEMCACHE_RESPONSE` événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

MEMCACHE_REQUEST	MEMCACHE_RESPONSE
clientIsExternal	accessTime
clientZeroWnd	clientIsExternal
isBinaryProtocol	clientZeroWnd
isNoReply	error
method	hits
receiverIsExternal	isBinaryProtocol
reqBytes	isRspImplicit
reqL2Bytes	method
reqPkts	misses
reqRTO	receiverIsExternal
reqSize	roundTripTime

MEMCACHE_REQUEST	MEMCACHE_RESPONSE
senderIsExternal	rspBytes
serverIsExternal	rspL2Bytes
serverZeroWnd	rspPkts
vbucket	rspRTO
	senderIsExternal
	serverIsExternal
	serverZeroWnd
	statusCode
	vbucket

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

`reqKeys:` ***Array***

Un tableau contenant les chaînes de clés Memcache envoyées avec la demande.

La valeur du `reqKeys` la propriété est la même lorsqu'on y accède sur l'un ou l'autre `MEMCACHE_REQUEST` ou le `MEMCACHE_RESPONSE` événement.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

Accès uniquement sur `MEMCACHE_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`reqSize:` ***Numéro***

Le nombre d'octets de requête L7, à l'exclusion des en-têtes Memcache. La valeur est `NaN` pour les requêtes sans charge utile, telles que GET et DELETE.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris les surcharges liées au protocole, telles que les en-têtes.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

Accès uniquement sur MEMCACHE_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`statusCode:` ***Corde***

Le code d'état de Memcache. Pour le protocole binaire, les métriques du système ExtraHop ajoutent à la méthode des codes d'état autres que NO_ERROR, mais pas la propriété StatusCode. Reportez-vous aux exemples de code correspondant au comportement des métriques du système ExtraHop.

Accès uniquement sur MEMCACHE_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`vbucket:` ***Numéro***

Le compartiment Memcache, s'il est disponible (protocole binaire uniquement).

Exemples de déclencheurs

- Exemple : enregistrer les succès et les échecs de Memcache
- Exemple : analyse des clés de cache mémoire

`Modbus`

Le `Modbus` la classe vous permet d'accéder aux propriétés depuis MODBUS_REQUEST et MODBUS_RESPONSE événements. Modbus est un protocole de communication série qui permet de connecter plusieurs appareils sur le même réseau .

Évènements

`MODBUS_REQUEST`

Fonctionne sur chaque requête envoyée par un client Modbus. Un client Modbus du système ExtraHop est l'équipement principal Modbus.

`MODBUS_RESPONSE`

Fonctionne sur chaque réponse envoyée par un serveur Modbus. Un serveur Modbus du système ExtraHop est l'équipement esclave Modbus.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un MODBUS_RESPONSE événement. Enregistrer les validations sur MODBUS_REQUEST les événements ne sont pas pris en charge.

Pour consulter les propriétés par défaut attribuées à l'objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`error:` ***Corde***

Le message d'erreur détaillé enregistré par le système ExtraHop.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`functionId:` ***Numéro***

Le code de fonction Modbus contenu dans la demande ou la réponse.

ID de fonction	Nom de la fonction
1	Read Coil
2	Read Discrete Inputs
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Holding Register
15	Write Multiple Coils
16	Write Multiple Holding Registers

`functionName:` ***Corde***

Nom du code de fonction Modbus contenu dans la demande ou la réponse.

`isReqAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la demande ne soit terminée.

`isRspAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la réponse ne soit terminée.

Accès uniquement sur `MODBUS_RESPONSE` événements ; sinon, une erreur se produira.

`payload:` ***Tampon***

Le Tampon objet contenant le corps de la demande ou de la réponse.

`payloadOffset:` ***Numéro***

Le décalage du fichier, exprimé en octets, dans le `resource` propriété. La propriété de charge utile est obtenue à partir du `resource` propriété au décalage.

`processingTime:` ***Numéro***

Le temps de traitement du serveur Modbus, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `MODBUS_RESPONSE` événements ; sinon, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `Modbus.commitRecord` sur un `MODBUS_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `error`
- `functionId`
- `functionName`
- `protocolId`
- `reqL2Bytes`
- `rspL2Bytes`
- `receiverIsExternal`
- `reqPkts`
- `rspPkts`
- `reqBytes`
- `rspBytes`
- `reqRTO`
- `rspRTO`

- roundTripTime
- clientZeroWnd
- senderIsExternal
- serverIsExternal
- serverZeroWnd
- statusCode
- txId
- unitId

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre d'octets de requête L2, y compris L2 en-têtes.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`reqPkts:` ***Numéro***

Le nombre de paquets contenus dans la demande.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`reqRTO:` ***Numéro***

Le nombre de délais de retransmission (RTOS) dans la demande.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`reqSize:` ***Numéro***

Le nombre d'octets de requête L7, à l'exclusion des en-têtes Modbus.

`reqTransferTime:` ***Numéro***

Le temps de transfert de la demande, exprimé en millisecondes. Si la demande est contenue dans un seul paquet, le temps de transfert est nul. Si la demande couvre plusieurs paquets, la valeur est le délai entre la détection du premier paquet de demande et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une demande importante ou un retard du réseau. La valeur est `NaN` s'il n'y a pas de mesure valide ou si le chronométrage n'est pas valide.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur MODBUS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, comme les en-têtes.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets contenus dans la réponse.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`rspRTO:` ***Numéro***

Le nombre de délais de retransmission (RTOS) dans la réponse.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`rspSize:` ***Numéro***

Nombre d'octets de réponse L7, à l'exclusion des en-têtes du protocole Modbus.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`rspTransferTime:` ***Numéro***

Le temps de transfert de la réponse, exprimé en millisecondes. Si la réponse est contenue dans un seul paquet, le temps de transfert est nul. Si la réponse couvre plusieurs paquets, la valeur est le délai entre la détection du premier paquet de réponse et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une réponse importante ou un retard du réseau. La valeur est `NaN` s'il n'y a pas de mesure valide ou si le chronométrage n'est pas valide.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`statusCode:` ***Numéro***

Le code d'état numérique de la réponse.

Numéro de code de statut	Description du statut
1	Illegal Function
2	Illegal Data Address
3	Illegal Data Value
4	Slave Device Failure
5	Acknowledge
6	Slave Device Busy
7	Negative Acknowledge
8	Memory Parity Error
10	Gateway Path Unavailable
11	Gateway Target Device Failed to Respond

Accès uniquement sur MODBUS_RESPONSE événements ; sinon, une erreur se produira.

`txId:` ***Numéro***

Identifiant de transaction de la demande ou de la réponse.

`unitId:` ***Numéro***

Identifiant d'unité du serveur Modbus répondant au client Modbus.

MongoDB

Le MongoDB la classe vous permet de stocker des métriques et d'accéder aux propriétés sur MONGODB_REQUEST et MONGODB_RESPONSE événements.

Évènements

`MONGODB_REQUEST`

S'exécute sur chaque requête MongoDB traitée par l'équipement.

`MONGODB_RESPONSE`

S'exécute sur chaque réponse MongoDB traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `MONGODB_REQUEST` ou `MONGODB_RESPONSE` événement.

L' événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`collection:` ***Corde***

Nom de la collection de base de données spécifiée dans la demande en cours.

`database:` ***Corde***

L'instance de base de données MongoDB. Dans certains cas, par exemple lorsque les événements de connexion sont chiffrés, le nom de la base de données n'est pas disponible.

`error:` ***Corde***

Le message d'erreur détaillé enregistré par le système ExtraHop.

Accès uniquement sur `MONGODB_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`isReqAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la requête MongoDB ne soit terminée.

`isReqTruncated:` ***Booléen***

La valeur est `true` si la taille du ou des documents demandés est supérieure à la taille maximale du document de charge utile.

`isRspAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la réponse MongoDB ne soit terminée.

Accès uniquement sur `MONGODB_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`method:` ***Corde***

La méthode de base de données MongoDB (apparaît sous **Méthodes** dans l' interface utilisateur).

`opcode:` ***Corde***

Le code opérationnel MongoDB sur le protocole filaire, qui peut être différent de la méthode MongoDB utilisée.

`processingTime:` ***Numéro***

Le temps de traitement de la demande, exprimé en millisecondes (équivalent à `rspTimeToFirstByte - reqTimeToLastByte`). La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `MONGODB_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `MongoDB.commitRecord()` sur l'un ou l'autre `MONGODB_REQUEST` ou `MONGODB_RESPONSE` événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

MONGODB_REQUEST	MONGODB_RESPONSE
clientIsExternal	clientIsExternal
clientZeroWnd	clientZeroWnd
collection	collection
database	database
isReqAborted	error
isReqTruncated	isRspAborted
method	method
opcode	opcode
receiverIsExternal	processingTime
reqBytes	receiverIsExternal
reqL2Bytes	roundTripTime
reqPkts	rspBytes
reqRTO	rspL2Bytes
reqSize	rspPkts
reqTimeToLastByte	rspRTO
senderIsExternal	rspSize
serverIsExternal	rspTimeToFirstByte
serverZeroWnd	rspTimeToLastByte
user	senderIsExternal
	serverIsExternal
	serverZeroWnd
	user

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

`reqSize:` ***Numéro***

Le nombre d'octets de requête L7, à l'exclusion des en-têtes MongoDB.

`reqTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la demande, exprimé en millisecondes.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`request:` ***Array***

Un tableau d'objets JS analysés à partir de documents de charge utile de requêtes MongoDB. La taille totale du document est limitée à 4K.

Si les documents BSON sont tronqués, `isReqTruncated` le drapeau est fixé. Les valeurs tronquées sont représentées comme suit :

- Les valeurs de chaîne primitives telles que le code, le code avec portée et les données binaires sont partiellement extraites.
- Les objets et les tableaux sont partiellement extraits.
- Toutes les autres valeurs primitives telles que Numbers, Dates, RegExp, etc., sont remplacées par `null`.

Si aucun document n'est inclus dans la demande, un tableau vide est renvoyé.

La valeur du `request` la propriété est la même lorsqu'on y accède sur l'un ou l'autre `MONGODB_REQUEST` ou le `MONGODB_RESPONSE` événement.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris les surcharges liées au protocole, telles que les en-têtes.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

`rspSize:` ***Numéro***

Le nombre d'octets de réponse L7, à l'exclusion des en-têtes MongoDB.

Accès uniquement sur `MONGODB_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspTimeToFirstByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le premier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur `MONGODB_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur MONGODB_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`user:` ***Corde***

Le nom d'utilisateur, s'il est disponible. Dans certains cas, par exemple lorsque les événements de connexion sont chiffrés, le nom d'utilisateur n'est pas disponible.

MSMQ

Le MSMQ la classe vous permet de stocker des métriques et d'accéder aux propriétés sur MSMQ_MESSAGE événements.

Évènements

MSMQ_MESSAGE

Fonctionne sur chaque message utilisateur MSMQ traité par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un MSMQ_MESSAGE événement.

Pour consulter les propriétés par défaut attribuées à l'objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`adminQueue:` ***Corde***

Nom de la file d'administration du message.

`correlationId:` ***Tampon***

ID de corrélation du message généré par l'application.

`dstQueueMgr:` ***Corde***

Le courtier de messages de destination du message.

`isEncrypted:` ***Booléen***

La valeur est `true` si la charge utile est cryptée.

`label:` ***Corde***

L'étiquette ou la description du message.

`msgClass:` ***Corde***

Classe de message du message. Les valeurs suivantes sont valides :

- MQMSG_CLASS_NORMAL
- MQMSG_CLASS_ACK_REACH_QUEUE
- MQMSG_CLASS_NACK_ACCESS_DENIED
- MQMSG_CLASS_NACK_BAD_DST_Q
- MQMSG_CLASS_NACK_BAD_ENCRYPTION
- MQMSG_CLASS_NACK_BAD_SIGNATURE
- MQMSG_CLASS_NACK_COULD_NOT_ENCRYPT

- MQMSG_CLASS_NACK_HOP_COUNT_EXCEEDED
- MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_MSG
- MQMSG_CLASS_NACK_NOT_TRANSACTIONAL_Q
- MQMSG_CLASS_NACK_PURGED
- MQMSG_CLASS_NACK_Q_EXCEEDED_QUOTA
- MQMSG_CLASS_NACK_REACH_QUEUE_TIMEOUT
- MQMSG_CLASS_NACK_SOURCE_COMPUTER_GUID_CHANGED
- MQMSG_CLASS_NACK_UNSUPPORTED_CRYPTO_PROVIDER
- MQMSG_CLASS_ACK_RECEIVE
- MQMSG_CLASS_NACK_Q_DELETED
- MQMSG_CLASS_NACK_Q_PURGED
- MQMSG_CLASS_NACK_RECEIVE_TIMEOUT
- MQMSG_CLASS_NACK_RECEIVE_TIMEOUT_AT_SENDER
- MQMSG_CLASS_REPORT

`msgId:` ***Numéro***

ID du message MSMQ du message.

`payload:` ***Tampon***

Le corps du message MSMQ.

`priority:` ***Numéro***

Priorité du message. Il peut s'agir d'un nombre compris entre 0 et 7.

`queue:` ***Corde***

Nom de la file d'attente de destination du message.

`receiverBytes:` ***Numéro***

Le nombre de L4 octets du récepteur.

`receiverL2Bytes:` ***Numéro***

Le nombre de L2 octets du récepteur.

`receiverPkts:` ***Numéro***

Le nombre de paquets du récepteur.

`receiverRTO:` ***Numéro***

Le nombre de délais de retransmission (RTOS) depuis le récepteur.

`receiverZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par le récepteur.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `MSMQ.commitRecord()` sur un `MSMQ_MESSAGE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- adminQueue
- clientIsExternal
- dstQueueMgr
- isEncrypted
- label
- msgClass
- msgId
- priority
- queue
- receiverBytes
- receiverIsExternal

- receiverL2Bytes
- receiverPkts
- receiverRTO
- receiverZeroWnd
- responseQueue
- roundTripTime
- senderBytes
- senderIsExternal
- serverIsExternal
- senderL2Bytes
- senderPkts
- senderRTO
- serverZeroWnd
- srcQueueMgr

`responseQueue:` ***Corde***

Nom de la file de réponses du message.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`senderBytes:` ***Numéro***

Le numéro de l'expéditeur L4 octets.

`senderL2Bytes:` ***Numéro***

Le numéro de l'expéditeur L2 octets.

`senderPkts:` ***Numéro***

Le nombre de paquets de l'expéditeur.

`senderRTO:` ***Numéro***

Le nombre de délais de retransmission (RTOS) de l'expéditeur.

`senderZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par l'expéditeur.

`srcQueueMgr:` ***Corde***

Le courtier de messages source du message.

`NetFlow`

Le `NetFlow` l'objet de classe vous permet de stocker des métriques et d'accéder à des propriétés sur `NETFLOW_RECORD` événements.

Évènements

`NETFLOW_RECORD`

S'exécute à la réception d'un enregistrement de flux provenant d'un réseau de flux.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `NETFLOW_RECORD` événement.

Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

`findField(field:` ***Numéro*** `, enterpriseId:` ***Numéro*** `):` ***Corde*** | ***Numéro*** | ***Adresse IP*** | ***Tampon*** | ***Booléen***

Recherche l'enregistrement NetFlow et renvoie le champ spécifié. Renvoie une valeur nulle si le champ ne figure pas dans l'enregistrement. Si l'option `enterpriseId` argument est inclus, le champ spécifié est renvoyé uniquement si l'identifiant d'entreprise correspond, sinon la méthode renvoie une valeur nulle.

`hasField(field:` ***Numéro*** `):` ***Booléen***

Détermine si le champ spécifié se trouve dans l'enregistrement NetFlow.

Propriétés

`age:` ***Numéro***

Le temps écoulé, exprimé en secondes, entre `first` et `last` valeurs de propriété indiquées dans l'enregistrement NetFlow.

`deltaBytes:` ***Numéro***

Le nombre de L3 octets dans le flux depuis le dernier `NETFLOW_RECORD` événement.

`deltaPkts:` ***Numéro***

Le nombre de paquets dans le flux depuis le dernier `NETFLOW_RECORD` événement.

`dscp:` ***Numéro***

Le nombre représentant la dernière valeur du point de code de services différenciés (DSCP) du paquet de flux.

`dscpName:` ***Corde***

Le nom associé à la valeur DSCP du paquet de flux. Le tableau suivant répertorie les noms DSCP les plus connus :

Numéro	Nom
8	CS1
10	AF11
12	AF12
14	AF13
16	CS2
18	AF21
20	AF22
22	AF23
24	CS3
26	AF31
28	AF32
30	AF33
32	CS4
34	AF41
36	AF42

Numéro	Nom
38	AF43
40	CS5
44	VA
46	EF
48	CS6
56	CS7

Interface de sortie : ***Interface de flux***

Le FlowInterface objet qui identifie l' équipement de sortie.

domaines : ***Array***

Tableau d'objets contenant des champs d'informations contenus dans les paquets de flux. Chaque objet peut contenir les propriétés suivantes :

ID de champ : ***Numéro***

Numéro d'identification qui représente le type de champ.

Identifiant d'entreprise : ***Numéro***

Le numéro d'identification qui représente les informations spécifiques à l'entreprise.

tout d'abord : ***Numéro***

Le temps écoulé, exprimé en millisecondes, depuis l'époque du premier paquet du flux.

format : ***Corde***

Format de l'enregistrement NetFlow. Les valeurs valides sont `NetFlow v5`, `NetFlow v9`, et `IPFIX`.

Interface d'entrée : ***Interface de flux***

Le FlowInterface objet qui identifie l' équipement d'entrée.

Préférence IP : ***Numéro***

La valeur du champ de priorité IP associé au DSCP du paquet de flux.

ipproto : ***Corde***

Le protocole IP associé au flux, tel que TCP ou UDP.

dernier : ***Numéro***

Le temps écoulé, exprimé en millisecondes, depuis l'époque du dernier paquet du flux.

réseau : ***Réseau Flow***

Un objet qui identifie FlowNetwork et contient les propriétés suivantes :

identifiant : ***Corde***

L'identifiant du FlowNetwork.

adresse iPad : ***Adresse IP***

L'adresse IP du FlowNetwork.

Prochaine Hip : ***Adresse IP***

L'adresse IP du routeur du saut suivant.

Domaine d'observation : ***Numéro***

L'ID du domaine d'observation pour le modèle.

récepteur : ***Objet***

Un objet qui identifie le récepteur et qui contient les propriétés suivantes :

en tant que : ***Numéro***

Numéro de système autonome (ASN) de l'équipement de destination.

adresse iPad : ***Adresse IP***

L'adresse IP de l'équipement de destination.

Longueur du préfixe : ***Numéro***

Le nombre de bits du préfixe de l'adresse de destination.

port : ***Numéro***

Numéro de port TCP ou UDP de l'équipement de destination.

enregistrement : ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `NetFlow.commitRecord()` sur un `NETFLOW_RECORD` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- âge
- Le client est externe
- Nom DSCP
- octets delta
- Delta PTS
- Interface de sortie
- premier
- format
- Interface d'entrée
- dernier
- réseau
- Adresse réseau
- Suivant Hop
- proto
- Récepteur ADDR
- Récepteur ASN
- Le récepteur est externe
- Port du récepteur
- Longueur du préfixe du récepteur
- Adresse de l'expéditeur
- Expéditeur : SN
- L'expéditeur est externe
- Le serveur est externe
- Port de l'expéditeur
- Longueur du préfixe de l'expéditeur
- Nom du drapeau TCP
- Drapeaux TCP

expéditeur : ***Objet***

Un objet qui identifie l'expéditeur et qui contient les propriétés suivantes :

en tant que : ***Numéro***

Le numéro de système autonome (ASN) de l'équipement source.

adresse iPad : ***Adresse IP***

L'adresse IP de l'équipement source.

Longueur du préfixe : ***Numéro***

Le nombre de bits du préfixe de l'adresse source.

port : ***Numéro***

Numéro de port TCP ou UDP de la source d'équipement.

Noms des drapeaux TCP : ***Array***

Tableau de chaînes contenant les noms d'indicateurs TCP, tels que SYN ou ACK, présents dans les paquets de flux.

Drapeaux TCP : ***Numéro***

Le OU au niveau du bit de tous les indicateurs TCP définis sur le flux.

ID du modèle : ***Numéro***

L'ID du modèle auquel l'enregistrement fait référence. Les ID de modèle ne s'appliquent qu'aux enregistrements IPFIX et NetFlow v9.

jouets : ***Numéro***

Numéro de type de service (ToS) défini dans l'en-tête IP.

NFS

Le NFS la classe vous permet de stocker des métriques et d'accéder à des propriétés sur NFS_REQUEST et NFS_RESPONSE événements.

Évènements

NFS_REQUEST

S'exécute sur chaque requête NFS traitée par l'équipement.

NFS_RESPONSE

S'exécute sur chaque réponse NFS traitée par l'équipement.

Note: Le NFS_RESPONSE l'événement se déroule après chaque NFS_REQUEST événement, même si la réponse correspondante n'est jamais observée par le système ExtraHop .

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un NFS_RESPONSE événement. Enregistrez les validations le NFS_REQUEST les événements ne sont pas pris en charge.

Pour afficher les propriétés par défaut validées pour l' objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`accessTime:` ***Numéro***

Temps nécessaire au serveur pour accéder à un fichier sur disque, exprimé en millisecondes. Pour NFS, il s'agit du temps écoulé entre chaque commande READ et WRITE non pipelinée dans un flux NFS jusqu'à ce que la charge utile contenant la réponse soit enregistrée par le système ExtraHop. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide ou n'est pas applicable.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`authMethod:` ***Corde***

Méthode d'authentification des utilisateurs.

`error:` ***Corde***

Message d'erreur détaillé enregistré par le système ExtraHop.

Accès uniquement sur `NFS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`fileHandle:` ***Tampon***

L'identificateur de fichier renvoyé par le serveur lors des opérations LOOKUP, CREATE, SYMLINK, MKNOD, LINK ou READDIRPLUS.

`isCommandFileInfo:` ***Booléen***

La valeur est `true` pour les commandes d'informations sur les fichiers.

`isCommandRead:` ***Booléen***

La valeur est `true` pour les commandes READ.

`isCommandWrite:` ***Booléen***

La valeur est `true` pour les commandes WRITE.

`isRspAborted:` ***Booléen***

La valeur est vraie si la connexion est fermée avant la fin de la réponse.

Accès uniquement sur `NFS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`method:` ***Corde***

La méthode NFS. Les méthodes valides sont répertoriées sous la métrique NFS dans le système ExtraHop.

`offset:` ***Numéro***

Le décalage de fichier associé aux commandes NFS READ et WRITE.

Accès uniquement sur `NFS_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées et abandonnées ou si le timing n' est pas valide.

Accès uniquement sur `NFS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `NFS.commitRecord()` sur un `NFS_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `accessTime`
- `authMethod`
- `clientIsExternal`
- `clientZeroWnd`
- `error`
- `isCommandFileInfo`
- `isCommandRead`
- `isCommandWrite`
- `isRspAborted`
- `method`
- `offset`
- `processingTime`
- `receiverIsExternal`
- `renameDirChanged`
- `reqSize`
- `reqXfer`
- `resource`

- rspSize
- rspXfer
- senderIsExternal
- serverIsExternal
- serverZeroWnd
- statusCode
- txID
- user
- version

Accédez à l'objet d'enregistrement uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

renameDirChanged: ***Booléen***

La valeur est true si une demande de renommage de ressource inclut un déplacement de répertoire.

Accès uniquement sur NFS_REQUEST événements ; dans le cas contraire, une erreur se produira.

reqBytes: ***Numéro***

Le nombre de L4 octets de requête, à l'exclusion des en-têtes L4.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

reqL2Bytes: ***Numéro***

Le nombre de L2 octets de requête, y compris les en-têtes L2.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

reqPkts: ***Numéro***

Le nombre de paquets de requêtes.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

reqRTO: ***Numéro***

Le numéro de demande délais de retransmission (RTO).

Accès uniquement sur NFS_REQUEST événements ; dans le cas contraire, une erreur se produira.

reqSize: ***Numéro***

Nombre d'octets de requête L7, à l'exclusion des en-têtes NFS.

reqTransferTime: ***Numéro***

Le temps de transfert de la demande, exprimé en millisecondes. Si la demande est contenue dans un seul paquet, le temps de transfert est nul. Si la demande s'étend sur plusieurs paquets, la valeur est la durée entre la détection du premier paquet de requête NFS et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une demande NFS importante ou un retard réseau. La valeur est NaN s'il n'y a pas de mesure valide ou si le chronométrage n'est pas valide.

Accès uniquement sur NFS_REQUEST événements ; dans le cas contraire, une erreur se produira.

reqZeroWnd: ***Numéro***

Le nombre de fenêtres nulles dans la demande.

resource: ***Corde***

Le chemin et le nom de fichier, concaténés ensemble.

roundTripTime: ***Numéro***

Le temps d'aller-retour médian (RTT), exprimé en millisecondes. La valeur est NaN s'il n'y a pas d'échantillons RTT.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

rspBytes: ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspSize:` ***Numéro***

Nombre d'octets de réponse L7, à l'exclusion des en-têtes NFS.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspTransferTime:` ***Numéro***

Le temps de transfert de la réponse, exprimé en millisecondes. Si la réponse est contenue dans un seul paquet, le temps de transfert est nul. Si la réponse couvre plusieurs paquets, la valeur est la durée entre la détection du premier paquet de réponse NFS et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une réponse NFS importante ou un retard du réseau. La valeur est `NaN` s'il n'y a pas de mesure valide ou si le chronométrage n'est pas valide.

Accès uniquement sur NFS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`statusCode:` ***Corde***

Le code dquo NFS de la demande ou de la réponse.

`symlink:` ***Tampon | nul***

L'argument spécifié dans une requête NFS SYMLINK.

La valeur est nulle si cette propriété est accessible lors d'un événement autre que NFS_REQUEST ou si `NFS.method` n'est pas `SYMLINK`.

`txId:` ***Numéro***

L'ID de transaction.

`user:` ***Corde***

L'ID de l'utilisateur Linux, au format `uid:xxxx`.

`verifierMethod:` ***Corde***

Méthode de vérification de l'expéditeur de la demande.

`version:` ***Numéro***

La version NFS.

NMF

La classe NMF (NET Message Framing Protocol) vous permet de stocker des métriques et d'accéder à des propriétés sur NMF_RECORD événements.

Évènements

`NMF_RECORD`

S'exécute sur chaque enregistrement NMF traité par l'équipement.

Méthodes

`commitRecord()`: ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `NMF_RECORD` événement. Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez la propriété d'enregistrement ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`envelope`: ***Tampon***

Le Tampon objet qui contient les octets de charge utile du message.

`wireSize`: ***Numéro***

Longueur de l'enregistrement brut tel qu'il a été observé, exprimée en octets. Si l' enregistrement est compressé, cette propriété reflète la longueur de l' enregistrement compressé.

`mode`: ***Numéro***

Code numérique du mode de communication. Les codes suivants sont valides :

Code	Descriptif
1	Singleton Unsized
2	Duplex
3	Simplex
4	Taille d'un single

`via`: ***Corde***

L'URI auquel les messages suivants seront envoyés.

`version`: ***Corde***

Version du protocole NMF.

`NTLM`

Le `NTLM` la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `NTLM_MESSAGE` événements.

Évènements

`NTLM_MESSAGE`

S'exécute sur tous les messages NTLM traités par l'équipement.

Méthodes

`commitRecord()`: ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `NTLM_MESSAGE` événement.

Pour consulter les propriétés par défaut attribuées à l'objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu' une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`containsMIC:` ***Booléen***

La valeur est vraie si le message inclut un code d'intégrité du message (MIC) qui garantit que le message n'a pas été falsifié.

`challenge:` ***Corde***

La chaîne de hachage du défi codée en hexadécimal.

`domain:` ***Corde***

Le nom de domaine du client inclus dans le calcul du hachage du défi.

`flags:` ***Numéro***

L'OR au niveau du bit des drapeaux de négociation NTLM. Pour plus d'informations, consultez le Documentation NTLM sur le site Web de Microsoft.

`msgType:` ***Corde***

Type de message NTLM. Les types de messages suivants sont valides :

- `NTLM_AUTH`
- `NTLM_CHALLENGE`
- `NTLM_NEGOTIATE`

`ntlm2RspAVPairs:` ***Array***

Tableau d'objets contenant des paires attribut-valeur NTLM. Pour plus d'informations, consultez le Documentation NTLM sur le site Web de Microsoft.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `NTLM.commitRecord()` sur un `NTLM_MESSAGE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `challenge`
- `clientIsExternal`
- `domain`
- `flags`
- `l7proto`
- `msgType`
- `proto`
- `receiverAddr`
- `receiverIsExternal`
- `receiverPort`
- `senderAddr`
- `senderIsExternal`
- `senderPort`
- `serverIsExternal`
- `user`
- `windowsVersion`
- `workstation`

`rspVersion:` ***Corde***

Version de NTLM implémentée dans la réponse NTLM_AUTH. La valeur est `null` pour les messages non authentifiés. Les versions suivantes sont valides :

- `LM`
- `NTLMv1`
- `NTLMv2`

`user:` ***Corde***

Le nom d'utilisateur du client inclus dans le calcul du hachage du défi.

`windowsVersion:` ***Corde***

Version de Windows exécutée sur le client incluse dans le calcul du hachage du challenge.

`workstation:` ***Corde***

Nom du poste de travail client inclus dans le calcul du hachage du défi.

NTP

La classe NTP (Network Time Protocol) vous permet de stocker des métriques et d'accéder à des propriétés sur `NTP_MESSAGE` événements.

Évènements

`NTP_MESSAGE`

S'exécute sur chaque message NTP traité par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `NTP_MESSAGE` événement. Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu' une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`flags:` ***Numéro***

Représentation décimale de l'octet contenant des informations sur les indicateurs NTP. L'indicateur de saut est contenu dans les deux premiers bits de l' octet, la version NTP est contenue dans les trois bits suivants et le mode de fonctionnement du protocole NTP est contenu dans les trois derniers bits.

`leapIndicator:` ***Numéro***

Indique si une seconde supplémentaire sera ajoutée ou supprimée par rapport à la dernière minute de la journée sur l'horloge système. Les valeurs suivantes sont valides :

Valeur	Descriptif
`0`	Aucune seconde supplémentaire ne sera ajoutée ni supprimée.
`1`	Une seconde supplémentaire sera ajoutée à la dernière minute de la journée. La dernière minute comptera 61 secondes.
`2`	Une seconde supplémentaire sera supprimée à compter de la dernière minute de la journée. La dernière minute comptera 59 secondes.
`3`	Inconnu Les horloges ne sont pas synchronisées actuellement.

`mode:` ***Numéro***

L'identifiant numérique du mode de fonctionnement du protocole NTP.

`modeName:` ***Corde***

Nom du mode de fonctionnement du protocole NTP. Les valeurs suivantes sont valides :

Valeur	Identifiant numérique
`reserved`	0
`symmetric active`	1
`symmetric passive`	2
`client`	3
`server`	4
`broadcast`	5
`NTP control message`	6
`reserved for private use`	7

`originTimestamp:` ***Numéro***

Heure locale du client à laquelle il a envoyé la demande au serveur, exprimée en fractions de secondes depuis l'époque NTP.

`payload:` ***Tampon***

Le Tampon objet qui contient les octets de charge utile bruts du message NTP.

`poll:` ***Numéro***

Durée maximale d'attente du système entre les messages NTP, exprimée en fractions de secondes.

`precision:` ***Numéro***

La précision de l'horloge du système, exprimée en fractions de seconde.

`receiveTimestamp:` ***Numéro***

Heure locale du serveur à laquelle le serveur a reçu la demande du client, exprimée en fractions de secondes depuis l'époque NTP.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `NTP.commitRecord()` sur un `NTP_MESSAGE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `application`
- `extensionCount`
- `flowId`
- `modeName`
- `originTimestamp`
- `poll`
- `precision`
- `receiver`
- `receiverAddr`
- `receiverIsExternal`
- `receiverPort`
- `receiveTimestamp`
- `referenceId`
- `referenceIdCode`

- `referenceTimestamp`
- `rootDelay`
- `stratum`
- `sender`
- `senderAddr`
- `senderIsExternal`
- `senderPort`
- `transmitTimestamp`
- `version`
- `vlan`

`referenceId:` ***Numéro***

L'identifiant numérique du serveur ou de l'horloge de référence.

`referenceIdCode:` ***Corde | Null***

ID de chaîne du serveur ou de l'horloge de référence.

`referenceTimestamp:` ***Numéro***

Date à laquelle l'horloge du système a été réglée ou corrigée pour la dernière fois, exprimée en fractions de secondes depuis l'époque NTP.

`rootDelay:` ***Numéro***

Le délai aller-retour jusqu'à l'horloge de référence, exprimé en secondes.

`rootDispersion:` ***Numéro***

Erreur maximale par rapport à l'horloge de référence, exprimée en secondes.

`stratum:` ***Numéro***

Strate NTP de l'horloge système.

`transmitTimestamp:` ***Numéro***

Heure locale du serveur à laquelle le serveur a envoyé la réponse au client, exprimée en fractions de secondes depuis l'époque NTP.

`version:` ***Numéro***

Version du protocole NTP.

POP3

Le POP3 la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `POP3_REQUEST` et `POP3_RESPONSE` événements.

Évènements

`POP3_REQUEST`

S'exécute sur chaque demande POP3 traitée par l'équipement.

`POP3_RESPONSE`

S'exécute sur chaque réponse POP3 traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `POP3_RESPONSE` événement. Enregistrer les validations sur `POP3_REQUEST` les événements ne sont pas pris en charge.

Pour consulter les propriétés par défaut attribuées à l' objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`dataSize:` ***Numéro***

Taille du message, exprimée en octets.

Accès uniquement sur `POP3_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`error:` ***Corde***

Le message d'erreur détaillé enregistré par le système ExtraHop.

Accès uniquement sur `POP3_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`isEncrypted:` ***Booléen***

La valeur est `true` si la transaction est effectuée sur un serveur POP3 sécurisé.

`isReqAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la demande POP3 ne soit terminée.

`isRspAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant la fin de la réponse POP3.

Accès uniquement sur `POP3_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`method:` ***Corde***

La méthode POP3 telle que RETR ou DELE.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n' est pas valide.

Accès uniquement sur `POP3_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`recipientList:` ***Array***

Tableau contenant une liste d'adresses de destinataires.

Accès uniquement sur `POP3_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `POP3.commitRecord()` sur un `POP3_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `clientZeroWnd`
- `dataSize`
- `error`
- `isEncrypted`
- `isReqAborted`
- `isRspAborted`
- `method`
- `processingTime`
- `receiverIsExternal`
- `recipientList`
- `reqSize`
- `reqTimeToLastByte`
- `rspSize`
- `rspTimeToFirstByte`

- rspTimeToLastByte
- sender
- senderIsExternal
- serverIsExternal
- serverZeroWnd
- statusCode

Accédez à l'objet d'enregistrement uniquement sur POP3_RESPONSE événements ; dans le cas contraire, une erreur se produira.

reqBytes: ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

reqL2Bytes: ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

reqPkts: ***Numéro***

Le nombre de paquets de demandes.

reqRTO: ***Numéro***

Le numéro de demande délais de retransmission (RTO).

reqSize: ***Numéro***

Nombre d'octets de requête L7, à l'exclusion des en-têtes POP3.

reqTimeToLastByte: ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la demande, exprimé en millisecondes. La valeur est NaN sur les demandes et réponses expirées, ou si le délai n'est pas valide.

reqZeroWnd: ***Numéro***

Le nombre de fenêtres nulles dans la demande.

roundTripTime: ***Numéro***

Le temps TCP aller-retour (RTT) médian, exprimé en millisecondes. La valeur est NaN s'il n'y a pas d'échantillons RTT.

Accès uniquement sur POP3_RESPONSE événements ; dans le cas contraire, une erreur se produira.

rspBytes: ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur POP3_RESPONSE événements ; dans le cas contraire, une erreur se produira.

rspL2Bytes: ***Numéro***

Le nombre de L2 octets de réponse, y compris les surcharges liées au protocole, telles que les en-têtes.

Accès uniquement sur POP3_RESPONSE événements ; dans le cas contraire, une erreur se produira.

rspPkts: ***Numéro***

Le nombre de paquets de réponse.

Accès uniquement sur POP3_RESPONSE événements ; dans le cas contraire, une erreur se produira.

rspRTO: ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

Accès uniquement sur POP3_RESPONSE événements ; dans le cas contraire, une erreur se produira.

rspSize: ***Numéro***

Nombre d'octets de réponse L7, à l'exclusion des en-têtes POP3.

Accès uniquement sur POP3_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspTimeToFirstByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le premier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur `POP3_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur `POP3_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`sender:` ***Corde***

Adresse de l'expéditeur du message.

Accès uniquement sur `POP3_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`status:` ***Corde***

Le message d'état POP3 de la réponse, qui peut être `OK`, `ERR` ou `NULL`.

Accès uniquement sur `POP3_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`QUIC`

Le `QUIC` la classe vous permet de stocker des métriques et d'accéder à des propriétés sur `QUIC_OPEN` et `QUIC_CLOSE` événements.

Évènements

`QUIC_CLOSE`

S'exécute lorsqu'une connexion QUIC est fermée.

`QUIC_OPEN`

S'exécute lorsqu'une connexion QUIC est ouverte.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur `QUIC_OPEN` ou `QUIC_CLOSE` événement. Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `QUIC.commitRecord()` sur l'un ou l'autre `QUIC_OPEN` ou `QUIC_CLOSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientAddr`
- `clientIsExternal`

- clientPort
- proto
- receiverIsExternal
- senderIsExternal
- serverAddr
- serverIsExternal
- serverPort
- sni
- version
- vlan

`sni:` ***Corde***

L'indication du nom du serveur (SNI), qui identifie le nom du serveur auquel le client se connecte.

`version:` ***Corde***

Version du protocole QUIC.

RDP

Le RDP (Remote Desktop Protocol) est un protocole propriétaire créé par Microsoft qui permet à un ordinateur Windows de se connecter à un autre ordinateur Windows sur le même réseau ou via Internet . Le RDP la classe vous permet de stocker des métriques et d'accéder à des propriétés sur RDP_OPEN, RDP_CLOSE, ou RDP_TICK événements.

Évènements

`RDP_CLOSE`

S'exécute lorsqu'une connexion RDP est fermée.

`RDP_OPEN`

S'exécute lorsqu'une nouvelle connexion RDP est ouverte.

`RDP_TICK`

S'exécute périodiquement pendant que l'utilisateur interagit avec l'application RDP.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un RDP_OPEN, RDP_CLOSE, ou RDP_TICK événement.

L'événement détermine quelles propriétés sont validées pour l'objet d'enregistrement. Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`clientBuild:` ***Corde***

Le numéro de version du client RDP. Cette propriété n'est pas disponible si la connexion RDP est cryptée.

`clientName:` ***Corde***

Le nom de l'ordinateur client. Cette propriété n'est pas disponible si la connexion RDP est cryptée.

`cookie:` ***Corde***

Le cookie de connexion automatique stocké par le client RDP.

`desktopHeight:` ***Numéro***

Hauteur du bureau, exprimée en pixels. Cette propriété n'est pas disponible si la connexion RDP est cryptée.

`desktopWidth:` ***Numéro***

Largeur du bureau, exprimée en pixels. Cette propriété n'est pas disponible si la connexion RDP est cryptée.

`encryptionProtocol:` ***Corde***

Le protocole avec lequel la transaction est cryptée.

`error:` ***Corde***

Message d'erreur détaillé enregistré par le système ExtraHop.

`isDecrypted:` ***Booléen***

La valeur est vraie si le système ExtraHop a déchiffré et analysé la transaction de manière sécurisée. L'analyse du trafic déchiffré peut révéler des menaces avancées qui se cachent dans le trafic chiffré.

`isEncrypted:` ***Booléen***

La valeur est `true` si la connexion RDP est cryptée.

`isError:` ***Booléen***

La valeur est `true` si une erreur s'est produite lors de l'événement.

`keyboardLayout:` ***Corde***

La disposition du clavier, qui indique la disposition des touches et la langue de saisie. Cette propriété n'est pas disponible si la connexion RDP est cryptée.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `RDP.commitRecord()` sur l'un ou l'autre `RDP_OPEN`, `RDP_CLOSE`, ou `RDP_TICK` événement.

L' objet d'enregistrement par défaut peut contenir les propriétés suivantes :

`RDP_OPEN` et `RDP_CLOSE`	`RDP_TICK`
`clientBuild`	`clientBuild`
`clientIsExternal`	`clientBytes`
`clientName`	`clientIsExternal`
`cookie`	`clientL2Bytes`
`desktopHeight`	`clientName`
`desktopWidth`	`clientPkts`
`error`	`clientRTO`
`isEncrypted`	`clientZeroWnd`
`keyboardLayout`	`cookie`
`receiverIsExternal`	`desktopHeight`
`requestedColorDepth`	`desktopWidth`
`requestedProtocols`	`error`
`selectedProtocol`	`isEncrypted`

RDP_OPEN et RDP_CLOSE	RDP_TICK
senderIsExternal	keyboardLayout
serverIsExternal	receiverIsExternal
	requestedColorDepth
	requestedProtocols
	roundTripTime
	selectedProtocol
	senderIsExternal
	serverBytes
	serverIsExternal
	serverL2Bytes
	serverPkts
	serverRTO
	serverZeroWnd

`requestedColorDepth:` ***Corde***

Profondeur de couleur demandée par le client RDP. Cette propriété n'est pas disponible si la connexion RDP est cryptée.

`requestedProtocols:` ***Tableau de cordes***

La liste des protocoles de sécurité pris en charge.

`reqBytes:` ***Numéro***

Le nombre de L4 octets dans la requête.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets dans la requête.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`reqPkts:` ***Numéro***

Le nombre de paquets contenus dans la demande.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`reqRTO:` ***Numéro***

Le nombre de délais de retransmission (RTO) dans la demande.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`roundTripTime:` ***Numéro***

Temps médian aller-retour (RTT) pendant la durée de l'événement, exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets contenus dans la réponse.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`rspRTO:` ***Numéro***

Le nombre de délais de retransmission (RTO) dans la réponse.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

Accès uniquement sur `RDP_TICK` événements ; sinon, une erreur se produira.

`selectedProtocol:` ***Corde***

Le protocole de sécurité sélectionné.

utilisateur : String

Le nom d'utilisateur, s'il est disponible. Dans certains cas, par exemple lorsque les événements de connexion sont chiffrés et que la sonde n'a pas été configurée pour décrypter le trafic, le nom d'utilisateur n'est pas disponible.

`Redis`

Remote Dictionary Server (Redis) est un serveur de structure de données en mémoire open-source. Le `Redis` la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `REDIS_REQUEST` et `REDIS_RESPONSE` événements.

Évènements

`REDIS_REQUEST`

Fonctionne sur chaque demande Redis traitée par l'équipement.

`REDIS_RESPONSE`

Fonctionne sur chaque réponse Redis traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `REDIS_REQUEST` ou `REDIS_RESPONSE` événement.

L'événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`errors:` ***Array***

Un ensemble de messages d'erreur détaillés enregistrés par le système ExtraHop.

Accès uniquement sur `REDIS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`isReqAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la demande Redis ne soit terminée.

`isRspAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la réponse Redis ne soit terminée.

Accès uniquement sur `REDIS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`method:` ***Corde***

La méthode Redis telle que GET ou KEYS.

`payload:` ***Tampon***

Le corps de la réponse ou de la demande.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n' est pas valide.

Accès uniquement sur `REDIS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `Redis.commitRecord()` sur l'un ou l'autre `REDIS_REQUEST` ou `REDIS_RESPONSE` événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

REDIS_REQUEST	REDIS_RESPONSE
clientIsExternal	clientIsExternal
clientZeroWnd	clientZeroWnd
method	error
receiverIsExternal	method
reqKey	processingTime
reqSize	receiverIsExternal
reqTransferTime	reqKey
isReqAborted	rspSize
senderIsExternal	rspTransferTime
serverZeroWnd	isRspAborted
	rspTimeToFirstByte
	rspTimeToLastByte
	senderIsExternal
	serverIsExternal
	serverZeroWnd

`reqKey:` ***Array***

Un tableau contenant les chaînes clés Redis envoyées avec la demande.

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

`reqSize:` ***Numéro***

Le nombre d'octets de requête L7, à l'exclusion des en-têtes Redis.

`reqTransferTime:` ***Numéro***

Le temps de transfert de la demande, exprimé en millisecondes. Si la demande est contenue dans un seul paquet, le temps de transfert est nul. Si la demande couvre plusieurs paquets, la valeur est le délai entre la détection du premier paquet de demande Redis et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une demande Redis importante ou un retard du réseau. La valeur est `NaN` s'il n'y a pas de mesure valide ou si le chronométrage n'est pas valide.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Le temps TCP aller-retour (RTT) médian, exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris les surcharges liées au protocole, telles que les en-têtes.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

`rspTransferTime:` ***Numéro***

Le temps de transfert de réponse, exprimé en millisecondes. Si la réponse est contenue dans un seul paquet, le temps de transfert est nul. Si la réponse couvre plusieurs paquets, la valeur est le délai entre la détection du premier paquet de réponse Redis et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une réponse Redis importante ou un retard du réseau. La valeur est `NaN` s'il n'y a pas de mesure valide ou si le chronométrage n'est pas valide.

Accès uniquement sur `REDIS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspSize:` ***Numéro***

Le nombre d'octets de réponse L7, à l'exclusion des en-têtes Redis.

Accès uniquement sur `REDIS_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspTimeToFirstByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le premier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur REDIS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur REDIS_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

RFB

Le RFB la classe vous permet de stocker des métriques et d'accéder aux propriétés sur RFB_OPEN, RFB_CLOSE, et RFB_TICK événements.

Évènements

`RFB_CLOSE`

S'exécute lorsqu'une connexion RFB est fermée.

`RFB_OPEN`

S'exécute lorsqu'une nouvelle connexion RFB est ouverte.

`RFB_TICK`

Fonctionne périodiquement sur les flux RFB.

Méthodes

`commitRecord():` ***vide***

Valide un objet d'enregistrement dans l'espace de stockage des enregistrements. Pour consulter les propriétés par défaut attribuées à l'objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu' une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`authType:` ***Numéro***

Numéro correspondant au type de sécurité négocié par le client et le serveur.

Accès uniquement sur RFB_OPEN événements ; dans le cas contraire, une erreur se produira.

Type de sécurité	Numéro
Invalid	0
None	1
VNC Authentication	2
RealVNC	3-15
Tight	16
Ultra	17
TLS	18
VeNCrypt	19

Type de sécurité	Numéro
GTK-VNC SASL	20
MD5 hash authentication	21
Colin Dean xvp	22
RealVNC	128-255

`authResult:` ***Numéro***

Indique si l'authentification a réussi.

Valeur	Descriptif
0	Succeeded
1	Failed

`duration:` ***Numéro***

Durée de la session RFB, exprimée en secondes.

Accès uniquement sur `RFB_CLOSE` événements ; dans le cas contraire, une erreur se produira.

`error:` ***Corde***

Le message d'erreur détaillé enregistré par le système ExtraHop.

Accès uniquement sur `RFB_OPEN` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement validé dans l'espace de stockage des enregistrements via un appel à `RFB.commitRecord()`.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

RFB_OPEN	RFB_TICK	RFB_CLOSE
authType	clientIsExternal	clientIsExternal
authResult	reqBytes	duration
clientIsExternal	receiverIsExternal	receiverIsExternal
error	reqL2Bytes	senderIsExternal
receiverIsExternal	reqPkts	serverIsExternal
senderIsExternal	reqRTO	
serverIsExternal	reqZeroWnd	
version	roundTripTime	
	rspBytes	
	rspL2Bytes	
	rspPkts	
	rspRTO	
	rspZeroWnd	
	senderIsExternal	

RFB_OPEN	RFB_TICK	RFB_CLOSE
	serverIsExternal	

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`roundTripTime:` ***Numéro***

Le temps TCP aller-retour (RTT) médian, exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d' échantillons RTT.

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris les surcharges liées au protocole, telles que les en-têtes.

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

Accès uniquement sur `RFB_TICK` événements ; dans le cas contraire, une erreur se produira.

`version:` ***Corde***

Version du protocole RFB négociée par le client et le serveur.

Accès uniquement sur `RFB_OPEN` événements ; dans le cas contraire, une erreur se produira.

RPC

Le RPC la classe vous permet de stocker des mesures et d'accéder aux propriétés de l'activité MSRPC (Microsoft Remote Procedure Call) sur RPC_REQUEST et RPC_RESPONSE événements.

Évènements

RPC_REQUEST

S'exécute sur chaque requête RPC traitée par l'équipement.

RPC_RESPONSE

S'exécute sur chaque réponse RPC traitée par l'équipement.

Méthodes

commitRecord(): ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un RPC_REQUEST ou RPC_RESPONSE événement.

Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

authType: ***Corde***

Type de sécurité négocié par le client et le serveur. Les types suivants sont valides :

- DIGEST
- DPA
- GSS_KERBEROS
- GSS_SCHANNEL
- KRB5
- MSN
- MQ
- NONE
- NTLMSSP
- SEC_CHAN
- SPNEGO

Accès uniquement sur RPC_RESPONSE événements ; dans le cas contraire, une erreur se produira.

encryptionProtocol: ***Corde***

Le protocole avec lequel la transaction est cryptée.

interface: ***Corde***

Le nom de l'interface RPC, tel que `drsuapi` et `epmapper`.

interfaceGUID: ***Corde***

Le GUID de l'interface RPC. Le format du GUID inclut des tirets, comme illustré dans l'exemple suivant :

```
367abb81-9844-35f2-ad32-98f038001004
```

isEncrypted: ***Booléen***

La valeur est vraie si la charge utile est cryptée.

`isDecrypted:` ***Booléen***

La valeur est vraie si le système ExtraHop a déchiffré et analysé la transaction de manière sécurisée. L'analyse du trafic déchiffré peut révéler les menaces avancées qui se cachent dans le trafic chiffré.

`isNDR64:` ***Booléen | nul***

Indique si la demande ou la réponse a été transmise avec la syntaxe de transfert NDR64. Si le `pduType` la propriété n'est pas une demande ou une réponse, la valeur est nulle.

`operation:` ***Corde***

Le nom de l'opération RPC, tel que `DRSGetNCChanges` et `ept_map`.

`opnum:` ***Numéro***

L'opnum de l'opération RPC. L'opnum est l'identifiant numérique de l'opération RPC.

`payload:` ***Tampon | nul***

Le Tampon objet contenant le corps de la demande ou de la réponse. Si le `pduType` la propriété n'est pas une demande ou une réponse, la valeur est nulle.

`pduType:` ***Corde***

Type de PDU, qui indique l'objectif du message RPC. Les valeurs suivantes sont valides :

- `ack`
- `alter_context`
- `alter_context_resp`
- `auth`
- `bind`
- `bind_ack`
- `bind_nak`
- `cancel_ack`
- `cl_cancel`
- `co_cancel`
- `fack`
- `fault`
- `nocall`
- `orphaned`
- `ping`
- `response`
- `request`
- `reject`
- `shutdown`
- `working`

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `RPC.commitRecord()` sur un `RPC_REQUEST` ou `RPC_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientAddr`
- `clientBytes`
- `clientIsExternal`
- `clientL2Bytes`
- `clientPkts`
- `clientPort`
- `clientRTO`
- `clientZeroWnd`
- `interface`

- operation
- proto
- receiverIsExternal
- roundTripTime
- senderIsExternal
- serverAddr
- serverBytes
- serverIsExternal
- serverL2Bytes
- serverPkts
- serverPort
- serverRTO
- serverZeroWnd

`reqBytes:` ***Numéro***

Le nombre de L4 octets de requête, à l'exclusion des en-têtes L4.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de requête, y compris les en-têtes L2.

`reqPkts:` ***Numéro***

Le nombre de paquets de requêtes.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Temps d'aller-retour (RTT) TCP médian, exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`RTCP`

Le RTCP la classe vous permet de stocker des métriques et d'accéder à des propriétés sur `RTCP_MESSAGE` événements.

Évènements

`RTCP_MESSAGE`

S'exécute sur chaque paquet UDP RTCP traité par l'équipement.

Méthodes

`commitRecord()`: ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `RTCP_MESSAGE` événement.

Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`callId`: ***Corde***

L'ID d'appel à associer à un SIP flux.

`packets`: ***Array***

Tableau d'objets de paquets RTCP où chaque objet représente un paquet et contient un champ PacketType. Chaque objet possède des champs différents en fonction du type de message, comme décrit ci-dessous.

`packetType`: ***Corde***

Type de paquet. Si le type de paquet n'est pas reconnaissable, `packetType` sera « N inconnu » où N est le RTP valeur du type de paquet de contrôle.

Valeur	Tapez	Nom
194	SMPTETC	SMPTE time-code mapping
195	IJ	Extended inter-arrival jitter report
200	SR	sender report
201	RR	receiver report
202	SDES	source description
203	BYE	goodbye
204	APP	application-defined
205	RTPFB	Generic RTP Feedback
206	PSFB	Payload-specific
207	XR	extended report
208	AVB	AVB RTCP packet
209	RSI	Receiver Summary Information
210	TOKEN	Port Mapping
211	IDMS	IDMS Settings

La liste suivante décrit les champs pour chaque type d' objet de paquet :

APP

`name:` ***Corde***

Le nom choisi par la personne qui définit l'ensemble de paquets APP comme étant unique. Interprété comme quatre caractères ASCII distinguant les majuscules et minuscules.

`ssrc:` ***Numéro***

Le SSRC de l'expéditeur.

`value:` ***Tampon***

Les données facultatives dépendantes de l'application.

BYE

`packetType:` ***Numéro***

Contient le numéro 203 pour l'identifier comme étant un paquet RTCP BYE.

SR

`ntpTimestamp:` ***Numéro***

L'horodateur NTP, converti en millisecondes depuis l' époque (1er janvier 1970).

`reportBlocks:` ***Array***

Tableau d'objets de rapport contenant :

`fractionLost:` ***Numéro***

Nombre de 8 bits indiquant le nombre de paquets perdus divisé par le nombre de paquets attendus.

`jitter:` ***Numéro***

Estimation de la variance statistique du temps entre les arrivées des paquets de données RTP, exprimée en millisecondes.

`lastSR:` ***Numéro***

Les 32 bits du milieu du NTP_Timestamp reçus dans le cadre du dernier paquet RTCP sender report (SR) du SSRC source. Si aucun SR n'a encore été reçu, ce champ est mis à zéro.

`lastSRDelay:` ***Numéro***

Le délai entre la réception du dernier paquet SR de la source SSRC et l'envoi de ce bloc de réception, exprimé en unités de 1/65536 secondes. Si aucun paquet SR n'a encore été reçu, ce champ est mis à zéro.

`packetsLost:` ***Numéro***

Nombre total de paquets de données RTP provenant de la source SSRC qui ont été perdus depuis le début de la réception.

`seqNum:` ***Numéro***

Le numéro de séquence le plus élevé reçu de la source SSRC.

`ssrc:` ***Numéro***

Le SSRC de l'expéditeur.

`rtpTimestamp:` ***Numéro***

L'horodateur RTP, converti en millisecondes depuis l'époque (1er janvier 1970).

`senderOctets:` ***Numéro***

Le nombre d'octets de l'expéditeur.

`senderPkts:` ***Numéro***

Le nombre de paquets de l'expéditeur.

RR

reportBlocks: ***Array***

Tableau d'objets de rapport contenant :

fractionLost: ***Numéro***

Nombre de 8 bits indiquant le nombre de paquets divisé pour la dernière fois par le nombre de paquets attendus.

jitter: ***Numéro***

Estimation de la variance statistique de l' intervalle entre les arrivées de paquets de données RTP, exprimée en millisecondes.

lastSR: ***Numéro***

Les 32 bits du milieu du NTP_Timestamp reçus dans le cadre du dernier paquet RTCP sender report (SR) du SSRC source. Si aucun SR n'a encore été reçu, ce champ est mis à zéro.

lastSRDelay: ***Numéro***

Le délai entre la réception du dernier paquet SR de la source SSRC et l'envoi de ce bloc de rapport de réception, exprimé en unités de 1/65536 secondes. Si aucun paquet SR n'a encore été reçu, ce champ est mis à zéro.

packetsLost: ***Numéro***

Nombre total de paquets de données RTP provenant de la source SSRC qui ont été perdus depuis le début de la réception.

seqNum: ***Numéro***

Le numéro de séquence le plus élevé reçu de la source SSRC.

ssrc: ***Numéro***

Le SSRC de l'expéditeur.

ssrc: ***Numéro***

Le SSRC de l'expéditeur.

SDES

descriptionBlocks: ***Array***

Tableau d'objets contenant :

type: ***Numéro***

Le type SDES.

Type de SDES	Abréviation.	Nom
0	END	end of SDES list
1	CNAME	canonical name
2	NAME	user name
3	EMAIL	user's electronic mail address
4	PHONE	user's phone number
5	LOC	geographic user location
6	TOOL	name of application or tool

Type de SDES	Abréviation.	Nom
7	NOTE	notice about the source
8	PRIV	private extensions
9	H323-C ADDR	H.323 callable address
10	APSI	Application Specific Identifier

`value:` ***Tampon***

Une mémoire tampon contenant la partie texte du paquet SDES.

`ssrc:` ***Numéro***

Le SSRC de l'expéditeur.

`XR`

`ssrc:` ***Numéro***

Le SSRC de l'expéditeur.

`xrBlocks:` ***Array***

Tableau de blocs de rapports contenant :

`statSummary:` ***Objet***

Type 6 uniquement. Le `statSummary` l'objet contient les propriétés suivantes :

`beginSeq:` ***Numéro***

Numéro de séquence de début de l'intervalle.

`devJitter:` ***Numéro***

L'écart type du temps de transit relatif entre chaque série de deux paquets dans l' intervalle de séquence.

`devTTLOrHL:` ***Numéro***

Écart type des valeurs TTL ou Hop Limit des paquets de données compris dans la plage de numéros de séquence.

`dupPackets:` ***Numéro***

Le nombre de paquets dupliqués dans l' intervalle des numéros de séquence.

`endSeq:` ***Numéro***

Le numéro de séquence de fin de l'intervalle.

`lostPackets:` ***Numéro***

Le nombre de paquets perdus dans l' intervalle des numéros de séquence.

`maxJitter:` ***Numéro***

Temps de transmission relatif maximal entre deux paquets dans l'intervalle de séquence, exprimé en millisecondes.

`maxTTLOrHL:` ***Numéro***

Valeur TTL ou limite de sauts maximale des paquets de données dans la plage de numéros de séquence.

`meanJitter:` ***Numéro***

Temps de transit relatif moyen entre deux séries de paquets dans l'intervalle de séquence, arrondi à la valeur la plus proche

pouvant être exprimée sous forme d'horodateur RTP, exprimé en millisecondes.

`meanTTLOrHL:` ***Numéro***

Valeur TTL ou limite de sauts moyenne des paquets de données compris dans la plage de numéros de séquence.

`minJitter:` ***Numéro***

Temps de transmission relatif minimal entre deux paquets dans l'intervalle de séquence, exprimé en millisecondes.

`minTTLOrHL:` ***Numéro***

Valeur TTL ou limite de sauts minimale des paquets de données compris dans la plage de numéros de séquence.

`ssrc:` ***Numéro***

Le SSRC de l'expéditeur.

`type:` ***Numéro***

Type de bloc XR.

Type de bloc	Nom
1	Loss RTE Report Block
2	Duplicate RLE Report Block
3	Packet Receipt Times Report Block
4	Receiver Reference Time Report Block
5	DLRR Report Block
6	Statistics Summary Report Block
7	VoIP Metrics Report Block
8	RTCP XP
9	Texas Instruments Extended VoIP Quality Block
10	Post-repair Loss RLE Report Block
11	Multicast Acquisition Report Block
12	IBMS Report Block
13	ECN Summary Report
14	Measurement Information Block
15	Packet Delay Variation Metrics Block
16	Delay Metrics Block
17	Burst/Gap Loss Summary Statistics Block
18	Burst/Gap Discard Summary Statistics Block

Type de bloc	Nom
19	Frame Impairment Statistics Summary
20	Burst/Gap Loss Metrics Block
21	Burst/Gap Discard Metrics Block
22	MPEG2 Transport Stream PSI-Independent Decodability Statistics Metrics Block
23	De-Jitter Buffer Metrics Block
24	Discard Count Metrics Block
25	DRLE (Discard RLE Report)
26	BDR (Bytes Discarded Report)
27	RFISD (RTP Flows Initial Synchronization Delay)
28	RFSO (RTP Flows Synchronization Offset Metrics Block)
29	MOS Metrics Block
30	LCB (Loss Concealment Metrics Block)
31	CSB (Concealed Seconds Metrics Block)
32	MPEG2 Transport Stream PSI Decodability Statistics Block

`typeSpecific:` ***Numéro***

Le contenu de ce champ dépend du type de bloc.

`value:` ***Tampon***

Le contenu de ce champ dépend du type de bloc.

`voipMetrics:` ***Objet***

Type 7 uniquement. Le `voipMetrics` l'objet contient les propriétés suivantes :

`burstDensity:` ***Numéro***

Fraction de paquets de données RTP perdus ou rejetés en rafale depuis le début de la réception.

`burstDuration:` ***Numéro***

Durée moyenne, exprimée en millisecondes, des périodes de rafale survenues depuis le début de la réception.

`discardRate:` ***Numéro***

Fraction de paquets de données RTP provenant de la source qui ont été rejetés depuis le début de la réception, en raison d'une arrivée tardive ou anticipée, d'une sous-utilisation ou d'un débordement de la mémoire tampon de gigue de réception.

`endSystemDelay:` ***Numéro***

Délai du système final estimé le plus récemment, exprimé en millisecondes.

`extRFactor:` ***Numéro***

La métrique de qualité du facteur R externe. La valeur 127 indique que ce paramètre n'est pas disponible.

`gapDensity:` ***Numéro***

Fraction de paquets de données RTP perdus ou rejetés dans les intervalles entre les rafales depuis le début de la réception.

`gapDuration:` ***Numéro***

Durée moyenne des périodes d'intervalle survenues depuis le début de la réception, exprimée en millisecondes.

`gmin:` ***Numéro***

Le seuil d'écart.

`jbAbsMax:` ***Numéro***

Délai maximal absolu, exprimé en millisecondes, que le tampon de gigue adaptatif peut atteindre dans les pires conditions.

`jbMaximum:` ***Numéro***

Le délai maximal actuel de la mémoire tampon de gigue, qui correspond au premier paquet arrivant qui ne serait pas rejeté, exprimé en millisecondes.

`jbNominal:` ***Numéro***

Le délai nominal actuel de la mémoire tampon de gigue, qui correspond au délai nominal de la mémoire tampon de gigue pour les paquets qui arrivent exactement à temps, exprimé en millisecondes.

`lossRate:` ***Numéro***

Fraction de paquets de données RTP provenant de la source perdus depuis le début de la réception.

`mosCQ:` ***Numéro***

Le score d'opinion moyen estimé pour la qualité conversationnelle (MOS-CQ). La valeur 127 indique que ce paramètre n' est pas disponible.

`mosLQ:` ***Numéro***

Le score d'opinion moyen estimé pour la qualité d'écoute (MOS-LQ). La valeur 127 indique que ce paramètre n'est pas disponible.

`noiseLevel:` ***Numéro***

Le niveau sonore, exprimé en décibels.

`rerl:` ***Numéro***

Valeur de perte de retour d'écho résiduelle, exprimée en décibels.

`rFactor:` ***Numéro***

La métrique de qualité du facteur R. La valeur 127 indique que ce paramètre n'est pas disponible.

`roundTripDelay:` ***Numéro***

Temps d'aller-retour (RTT) calculé le plus récemment entre les interfaces RTP, exprimé en millisecondes.

`rxConfig:` ***Numéro***

L'octet de configuration du récepteur.

`signalLevel:` ***Numéro***

Le niveau relatif du signal vocal, exprimé en décibels.

`ssrc:` ***Numéro***

Le SSRC de l'expéditeur.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `RTCP.commitRecord()` sur un `RTCP_MESSAGE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `callId`
- `clientIsExternal`
- `cName`
- `flowId`
- `receiverIsExternal`
- `senderIsExternal`
- `serverIsExternal`
- `signalingFlowId`

  L'ID du flux SIP ou SCCP correspondant, qui négocie l'appel VoIP surveillé par le flux RTCP.

`RTP`

Le RTP la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `RTP_OPEN`, `RTP_CLOSE`, et `RTP_TICK` événements.

Évènements

`RTP_CLOSE`

S'exécute lorsqu'une connexion RTP est fermée.

`RTP_OPEN`

S'exécute lorsqu'une nouvelle connexion RTP est ouverte.

`RTP_TICK`

S'exécute périodiquement sur les flux RTP.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `RTP_TICK` événement. Enregistrez les validations le `RTP_OPEN` et `RTP_CLOSE` les événements ne sont pas pris en charge.

Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`bytes:` ***Numéro***

Le nombre d'octets envoyés.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

`callId:` ***Corde***

L'ID d'appel associé au flux SIP ou SCCP.

`drops:` ***Numéro***

Le nombre de paquets abandonnés détectés.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

`dups:` ***Numéro***

Nombre de paquets dupliqués détectés.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

`jitter:` ***Numéro***

Estimation de la variance statistique du temps entre les arrivées des paquets de données.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

`l2Bytes:` ***Numéro***

Le nombre de L2 octets.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

`mos:` ***Numéro***

Le score d'opinion moyen estimé pour la qualité.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

`outOfOrder:` ***Numéro***

Le nombre de messages hors service détectés.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

`payloadType:` ***Corde***

Type de charge utile RTP.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

payloadTypeId	payloadType
0	ITU-T G.711 PCMU Audio
3	GSM 6.10 Audio
4	ITU-T G.723.1 Audio
5	IMA ADPCM 32kbit Audio
6	IMA ADPCM 64kbit Audio
7	LPC Audio
8	ITU-T G.711 PCMA Audio
9	ITU-T G.722 Audio
10	Linear PCM Stereo Audio
11	Linear PCM Audio
12	QCELP
13	Comfort Noise
14	MPEG Audio
15	ITU-T G.728 Audio
16	IMA ADPCM 44kbit Audio

payloadTypeId	payloadType
17	IMA ADPCM 88kbit Audio
18	ITU-T G.729 Audio
25	Sun CellB Video
26	JPEG Video
28	Xerox PARC Network Video
31	ITU-T H.261 Video
32	MPEG Video
33	MPEG-2 Transport Stream
34	ITU-T H.263-1996 Video

`payloadTypeId:` ***Numéro***

La valeur numérique du type de charge utile. Voir le tableau ci-dessous `payloadType`.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

`pkts:` ***Numéro***

Le nombre de paquets envoyés.

Accès uniquement sur `RTP_TICK` événements ; sinon, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `RTP.commitRecord()` sur un `RTP_TICK` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `bytes`
- `callId`
- `clientIsExternal`
- `drops`
- `dups`
- `flowId`
- `jitter`
- `l2Bytes`
- `mos`
- `outOfOrder`
- `payloadType`
- `payloadTypeId`
- `pkts`
- `receiverIsExternal`
- `rFactor`
- `senderIsExternal`
- `serverIsExternal`
- `signalingFlowId`

  L'ID du flux SIP ou SCCP correspondant, qui négocie l'appel VoIP diffusé par le flux RTP.
- `ssrc`
- `version`

Accédez aux objets d'enregistrement uniquement sur RTP_TICK événements ; sinon, une erreur se produira.

`rFactor:` ***Numéro***

La métrique de qualité du facteur R.

Accès uniquement sur RTP_TICK événements ; sinon, une erreur se produira.

`ssrc:` ***Numéro***

Le SSRC de l'expéditeur.

`version:` ***Numéro***

Le numéro de version du RTP.

SCCP

Le Skinny Client Control Protocol (SCCP) est un protocole propriétaire de Cisco pour communiquer avec les appareils VoIP. Le SCCP la classe vous permet de stocker des métriques et d'accéder aux propriétés sur SCCP_MESSAGE événements.

Évènements

SCCP_MESSAGE

Fonctionne sur tous les messages SCCP traités par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un SCCP_MESSAGE événement.

Pour consulter les propriétés par défaut attribuées à l'objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`callId:` ***Corde***

L'identifiant d'appel associé au RTP flux.

`callInfo:` ***Objet***

Un objet contenant des informations sur le SCCP actuellement appelé. L'objet contient les champs suivants :

`callReference:` ***Numéro***

Identifiant unique de l'appel.

`callType:` ***Numéro***

L'ID du type d'appel.

IDENTIFIANT	Type d'appel
1	Inbound
2	Outbound
3	Forward

`calledPartyName:` ***Corde***

Le nom du destinataire de l'appel.

`calledPartyNumber:` ***Corde***

Numéro de téléphone du destinataire de l'appel.

`callingPartyName:` ***Corde***

Le nom de l'appelant.

`callingPartyNumber:` ***Corde***

Le numéro de téléphone de l'appelant.

`lineInstance:` ***Numéro***

Identifiant unique de la ligne.

`callStats:` ***Objet***

Objet contenant les statistiques relatives à l'appel SCCP, telles que rapportées et calculées par le client. L'objet contient les champs suivants :

`reportedBytesIn:` ***Numéro***

Le nombre de L7 octets reçus.

`reportedBytesOut:` ***Numéro***

Le nombre de L7 octets envoyés.

`reportedJitter:` ***Numéro***

Le niveau de gigue du paquet, ou variation de latence, pendant l'appel.

`reportedLatency:` ***Numéro***

Le niveau de latence du paquet, exprimé en millisecondes, pendant l'appel.

`reportedPktsIn:` ***Numéro***

Le nombre de paquets reçus.

`reportedPktsLost:` ***Numéro***

Le nombre de paquets perdus pendant l'appel.

`reportedPktsOut:` ***Numéro***

Le nombre de paquets envoyés.

`msgType:` ***Corde***

Le type de message SCCP décodé.

`receiverBytes:` ***Numéro***

Le nombre de L4 octets du récepteur.

`receiverL2Bytes:` ***Numéro***

Le nombre de L2 octets du récepteur.

`receiverPkts:` ***Numéro***

Le nombre de paquets provenant du récepteur.

`receiverRTO:` ***Numéro***

Le nombre de délais de retransmission (RTOS) depuis le récepteur.

`receiverZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles provenant du récepteur.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `SCCP.commitRecord()` sur un `SCCP_MESSAGE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `msgType`

- receiverBytes
- receiverIsExternal
- receiverL2Bytes
- receiverPkts
- receiverRTO
- receiverZeroWnd
- roundTripTime
- senderBytes
- senderIsExternal
- senderL2Bytes
- senderPkts
- senderRTO
- senderZeroWnd
- serverIsExternal

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`senderBytes:` ***Numéro***

Le nombre de L4 octets de l'expéditeur.

`senderL2Bytes:` ***Numéro***

Le nombre de L2 octets de l'expéditeur.

`senderPkts:` ***Numéro***

Le nombre de paquets provenant de l'expéditeur.

`senderRTO:` ***Numéro***

Le nombre de délais de retransmission (RTOS) de l'expéditeur.

`senderZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles provenant de l'expéditeur.

SDP

Le SDP la classe vous permet d'accéder aux propriétés sur `SIP_REQUEST` et `SIP_RESPONSE` événements.

Le `SIP_REQUEST` et `SIP_RESPONSE` les événements sont définis dans le SIP section.

Propriétés

`mediaDescriptions:` ***Array***

Tableau d'objets contenant les champs suivants :

`attributes:` ***Tableau de chaînes***

Les attributs de session facultatifs.

`bandwidth:` ***Tableau de chaînes***

Type de bande passante proposé en option et bande passante à consommer par la session ou le média.

`connectionInfo:` ***Corde***

Les données de connexion, notamment le type de réseau, le type d'adresse et l'adresse de connexion. Peut également contenir des sous-champs facultatifs, selon le type d'adresse.

`description:` ***Corde***

Description de session qui peut contenir une ou plusieurs descriptions de médias. Chaque description multimédia comprend les champs du média, du port et du protocole de transport.

`encryptionKey:` ***Corde***

Méthode de chiffrement et clé facultatives pour la session.

`mediaTitle:` ***Corde***

Titre du flux multimédia.

`sessionDescription:` ***Objet***

Un objet qui contient les champs suivants :

`attributes:` ***Tableau de chaînes***

Les attributs de session facultatifs.

`bandwidth:` ***Tableau de chaînes***

Type de bande passante proposé en option et bande passante à consommer par la session ou le média.

`connectionInfo:` ***Corde***

Les données de connexion, notamment le type de réseau, le type d'adresse et l'adresse de connexion. Peut également contenir des sous-champs facultatifs, selon le type d'adresse.

`email:` ***Corde***

L'adresse e-mail facultative. S'il est présent, il peut contenir plusieurs adresses e-mail.

`encryptionKey:` ***Corde***

Méthode de chiffrement et clé facultatives pour la session.

`origin:` ***Corde***

L'initiateur de la session, y compris le nom d'utilisateur, l'adresse de l'hôte de l'utilisateur, un identifiant de session et un numéro de version.

`phoneNumber:` ***Corde***

Le numéro de téléphone optionnel. S'il est présent, il peut contenir plusieurs numéros de téléphone.

`sessionInfo:` ***Corde***

Description de la session.

`sessionName:` ***Corde***

Le nom de la session.

`timezoneAdjustments:` ***Corde***

Le temps de réglage et le décalage pour une session planifiée.

`uri:` ***Corde***

L'URI facultatif destiné à fournir plus d'informations sur la session.

`version:` ***Corde***

Le numéro de version. Cela devrait être 0.

`timeDescriptions:` ***Array***

Tableau d'objets contenant les champs suivants :

`repeatTime:` ***Corde***

Le temps de répétition de la session, y compris l'intervalle, la durée active et les décalages par rapport à l'heure de début.

`time:` ***Corde***

Heure de début et heure de fin d'une session.

`SFlow`

Le `SFlow` un objet de classe vous permet de stocker des métriques et d'accéder aux propriétés sur `SFLOW_RECORD` événements. sFlow est une technologie d'échantillonnage permettant de surveiller le trafic

dans les réseaux de données. sFlow échantillonne chaque nième paquet et l'envoie au collecteur tandis que NetFlow envoie les données de chaque flux au collecteur. La principale différence entre sFlow et NetFlow est que sFlow est indépendant de la couche réseau et peut échantillonner n'importe quoi.

Évènements

`SFLOW_RECORD`

S'exécute à la réception d'un échantillon sFlow exporté depuis un réseau de flux.

Méthodes

`commitRecord():` ***vide***

Envoie un objet d'enregistrement de flux, qui indique le format sFlow, à l'espace de stockage des enregistrements configuré sur un `SFLOW_RECORD` événement.

Pour afficher les propriétés par défaut attribuées à l'objet d'enregistrement, consultez la propriété d'enregistrement ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu' une seule fois, même si `.commitRecord` est appelé plusieurs fois pour le même enregistrement unique.

Propriétés

`deltaBytes:` ***Numéro***

Le nombre d'octets L3 dans le paquet de flux.

`dscp:` ***Numéro***

Numéro représentant la dernière valeur de point de code de services différenciés (DSCP) du paquet de flux.

`dscpName:` ***Corde***

Nom associé à la valeur DSCP transmise par un équipement dans le flux. Le tableau suivant affiche les noms DSCP connus :

Numéro	Nom
8	CS1
10	AF11
12	AF12
14	AF13
16	CS2
18	AF21
20	AF22
22	AF23
24	CS3
26	AF31
28	AF32
30	AF33
32	CS4
34	AF41
36	AF42

Numéro	Nom
38	AF43
40	CS5
44	VA
46	EF
48	CS6
56	CS7

`egressInterface:` ***Interface de flux***

Le FlowInterface objet identifiant l' interface de sortie.

`format:` ***Corde***

Format de l'enregistrement sFlow. La valeur valide est « sFlow v5 ».

`headerData:` ***Tampon***

Le Tampon objet contenant les octets bruts de l' intégralité de l'en-tête du paquet de flux.

`ingressInterface:` ***Interface de flux***

Le FlowInterface objet identifiant l' interface d'entrée.

`ipPrecedence:` ***Numéro***

La valeur du champ de priorité IP associé au DSCP du paquet de flux.

`ipproto:` ***Corde***

Le protocole IP associé au flux, tel que TCP ou UDP.

`network:` ***Réseau Flow***

Renvoie un FlowNetwork objet identifiant l' exportateur et contenant les propriétés suivantes :

`id:` ***Corde***

L'identifiant du FlowNetwork.

`ipaddr:` ***Adresse IP***

L'adresse IP du FlowNetwork.

`record:` ***Objet***

L'objet d'enregistrement de flux qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `SFlow.commitRecord()` sur un `SFLOW_RECORD` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `deltaBytes`
- `dscpName`
- `egressInterface`
- `format`
- `ingressInterface`
- `ipPrecedence`
- `ipproto`
- `network`
- `networkAddr`
- `receiverIsExternal`
- `senderIsExternal`
- `serverIsExternal`
- `tcpFlagName`

- tcpFlags

tcpFlagNames: ***Array***

Un tableau de chaînes contenant des noms d'indicateurs TCP, tels que SYN ou ACK, présent dans les paquets de flux.

tcpFlags: ***Numéro***

Le bit par bit OR de tous les indicateurs TCP définis sur le flux.

tos: ***Numéro***

Numéro de type de service (ToS) défini dans l'en-tête IP.

SIP

Le SIP la classe vous permet de stocker des métriques et d'accéder à des propriétés sur SIP_REQUEST et SIP_RESPONSE événements.

Évènements

SIP_REQUEST

S'exécute sur chaque requête SIP traitée par l'équipement.

SIP_RESPONSE

S'exécute sur chaque réponse SIP traitée par l'équipement.

Méthodes

commitRecord(): ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur SIP_REQUEST ou SIP_RESPONSE événement.

L'événement détermine quelles propriétés sont validées pour l'objet d'enregistrement. Pour consulter les propriétés par défaut validées pour chaque événement, consultez le record propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si commitRecord() La méthode est appelée plusieurs fois pour le même enregistrement unique.

findHeaders(name: ***Corde***): ***Array***

Permet d'accéder aux valeurs d'en-tête SIP. Le résultat est un tableau d'objets d'en-tête (avec des propriétés de nom et de valeur) dont les noms correspondent au préfixe de la chaîne transmise à findHeaders.

Propriétés

callId: ***Corde***

ID d'appel pour ce message.

from: ***Corde***

Le contenu de l'en-tête From.

hasSDP: ***Booléen***

La valeur est true si cet événement inclut SDP information.

headers: ***Objet***

Un objet semblable à un tableau qui permet d'accéder aux noms et aux valeurs des en-têtes SIP. Accédez à un en-tête spécifique à l'aide de l'une des méthodes suivantes :

propriété de chaîne :

Le nom de l'en-tête, accessible à la manière d'un dictionnaire. Par exemple :

```
var headers = SIP.headers;
session = headers["X-Session-Id"];
accept = headers.accept;
```

propriété numérique :

Ordre dans lequel les en-têtes apparaissent sur le fil. L'objet renvoyé possède un nom et une propriété value. Les propriétés numériques sont utiles pour effectuer des itérations sur tous les en-têtes et lever l'ambiguïté des en-têtes dont le nom est dupliqué. Par exemple :

```
for (i = 0; i < headers.length; i++) {
   hdr = headers[i];
   debug("headers[" + i + "].name: " + hdr.name);
   debug("headers[" + i + "].value: " + hdr.value);
}
```

Note: Épargner `SIP.headers` dans le magasin Flow n'enregistre pas toutes les valeurs d'en-tête individuelles. Il est recommandé d'enregistrer les valeurs d'en-tête individuelles dans le magasin Flow.

`method:` ***Corde***

La méthode SIP.

Nom de la méthode	Descriptif
ACK	Confirme le client a reçu une réponse finale à une demande INVITE.
BYE	Met fin à un appel. Peut être envoyé par l'appelant ou par l'appelé.
CANCEL	Annule toute demande en attente
INFO	Envoie des informations de mi-session qui ne modifient pas l'état de la session.
INVITE	Invite un client pour participer à une session d'appel.
MESSAGE	Transporte les messages instantanés à l'aide du protocole SIP.
NOTIFY	Avertissez l'abonné d'un nouvel événement.
OPTIONS	Interroge les capacités des serveurs.
PRACK	Accusé de réception provisoire.
PUBLISH	Publiez un événement sur le serveur.
REFER	Demandez au destinataire d'émettre une demande SIP (transfert d'appel).
REGISTER	Enregistre l'adresse répertoriée dans le champ d'en-tête To auprès d'un serveur SIP.
SUBSCRIBE	S'abonne à un événement de notification de la part du notifiant.
UPDATE	Modifie l'état d'une session sans modifier l'état de la boîte de dialogue.

`payload:` ***Tampon*** | ***nul***

Le Tampon objet qui contient les octets de charge utile bruts de la transaction d'événement. Si la charge utile a été compressée, le contenu décompressé est renvoyé.

La mémoire tampon contient *N* premiers octets de la charge utile, où *N* est le nombre d'octets de charge utile spécifié par Octets vers la mémoire tampon champ lorsque le déclencheur a été configuré via l'interface utilisateur Web ExtraHop. Le nombre d'octets par défaut est de 2 048. Pour plus d'informations, voir Options de déclencheur avancées.

`processingTime:` ***Numéro***

Le temps entre la demande et la première réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées et abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `SIP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `SIP.commitRecord()` sur l'un ou l'autre `SIP_REQUEST` ou `SIP_RESPONSE` événement.

L'événement pour lequel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

SIP_REQUEST	SIP_RESPONSE
callId	callId
clientIsExternal	clientIsExternal
clientZeroWnd	clientZeroWnd
from	from
hasSDP	hasSDP
method	processingTime
receiverIsExternal	receiverIsExternal
reqBytes	roundTripTime
reqL2Bytes	rspBytes
reqPkts	rspL2Bytes
reqRTO	rspPkts
reqSize	rspRTO
senderIsExternal	rspSize
serverIsExternal	senderIsExternal
serverZeroWnd	serverIsExternal
to	serverZeroWnd
uri	statusCode
	to

`reqBytes:` ***Numéro***

Le nombre de L4 octets de requête, à l'exclusion des en-têtes L4.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de requête, y compris les en-têtes L2.

`reqPkts:` ***Numéro***

Le nombre de paquets de requêtes.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

`reqSize:` ***Numéro***

Nombre d'octets de requête L7, à l'exclusion des en-têtes SIP.

Accès uniquement sur `SIP_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Le temps médian aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris la surcharge du protocole, telle que les en-têtes.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

`rspSize:` ***Numéro***

Nombre d'octets de réponse L7, à l'exclusion des en-têtes SIP.

Accès uniquement sur `SIP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`statusCode:` ***Numéro***

Le code d'état de la réponse SIP.

Accès uniquement sur `SIP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

Le tableau suivant affiche les réponses provisoires :

Numéro	Réponse
100	Trying
180	Ringing
181	Call is Being Forwarded
182	Queued
183	Session In Progress
199	Early Dialog Terminated

Le tableau suivant répertorie les réponses réussies :

Numéro	Réponse
200	OK

Numéro	Réponse
202	Accepted
204	No Notification

Le tableau suivant affiche les réponses de redirection :

Numéro	Réponse
300	Multiple Choice
301	Moved Permanently
302	Moved Temporarily
305	Use Proxy
380	Alternative Service

Le tableau suivant affiche client réponses en cas de défaillance :

Numéro	Réponse
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout
409	Conflict
410	Gone
411	Length Required
412	Conditional Request Failed
413	Request Entity Too Large
414	Request URI Too Long
415	Unsupported Media Type
416	Unsupported URI Scheme
417	Unknown Resource Priority
420	Bad Extension
421	Extension Required
422	Session Interval Too Small
423	Interval Too Brief

Numéro	Réponse
424	Bad Location Information
428	Use Identity Header
429	Provide Referrer Identity
430	Flow Failed
433	Anonymity Disallowed
436	Bad Identity Info
437	Unsupported Certificate
438	Invalid Identity Header
439	First Hop Lacks Outbound Support
470	Consent Needed
480	Temporarily Unavailable
481	Call/Transaction Does Not Exist
482	Loop Detected
483	Too Many Hops
484	Address Incomplete
485	Ambiguous
486	Busy Here
487	Request Terminated
488	Not Acceptable Here
489	Bad Event
491	Request Pending
493	Undecipherable
494	Security Agreement Required

Le tableau suivant répertorie les réponses aux pannes du serveur :

Numéro	Réponse
500	Server Internal Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Server Timeout
505	Version Not Supported
513	Message Too Large
580	Precondition Failure

Le tableau suivant affiche les réponses globales aux défaillances :

Nom	Réponse
600	Busy Everywhere
603	Decline
604	Does Not Exist Anywhere
606	Not Acceptable

`to:` ***Corde***

Le contenu de l'en-tête To.

`uri:` ***Corde***

L'URI pour SIP demande ou réponse.

SLP

Le SLP la classe vous permet de stocker des métriques et d'accéder aux propriétés sur SLP_MESSAGE événements.

Évènements

SLP_MESSAGE

Fonctionne sur tous les messages SLP traités par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un SLP_MESSAGE événement.

Pour consulter les propriétés par défaut validées, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement.

Propriétés

`attrList:` ***Corde | nul***

Les attributs du message SLP, dans une liste séparée par des virgules.

`functionId:` ***Numéro***

L'ID de fonction numérique du message SLP, qui correspond à la chaîne de type de message.

`msgType:` ***Corde***

Chaîne de type de message SLP, qui correspond à l'ID de fonction numérique, comme indiqué dans le tableau suivant :

Type de message	ID de fonction
Service Request	1
Service Reply	2
Service Registration	3
Service Deregister	4
Service Acknowledge	5
Attribute Request	6

Type de message	ID de fonction
Attribute Reply	7
DA Advertisement	8
Service Type Request	9
Service Type Reply	10
SA Advertisement	11

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `SLP.commitRecord()` lors d'un événement SLP_MESSAGE. L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `functionId`
- `msgType`
- `receiverIsExternal`
- `scopeList`
- `senderIsExternal`
- `serverIsExternal`

`scopeList:` ***Corde*** | ***nul***

L'étendue du message SLP, dans une liste séparée par des virgules.

SMPP

Le SMPP la classe vous permet de stocker des métriques et d'accéder aux propriétés sur SMPP_REQUEST et SMPP_RESPONSE événements.

Note: Le `mdn`, `shortcode`, et `error` les propriétés peuvent être `null`, en fonction de la disponibilité et de la pertinence.

Évènements

SMPP_REQUEST

S'exécute sur chaque demande SMPP traitée par l'équipement.

SMPP_RESPONSE

S'exécute sur chaque réponse SMPP traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un SMPP_RESPONSE événement. Enregistrer les validations sur SMPP_REQUEST les événements ne sont pas pris en charge.

Pour consulter les propriétés par défaut attribuées à l' objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`command:` ***Corde***

ID de commande SMPP.

`destination:` ***Corde***

L'adresse de destination telle que spécifiée dans le `SMPP_REQUEST`. La valeur est `null` si cela n'est pas disponible pour le type de commande actuel.

`error:` ***Corde***

Le code d'erreur correspondant à command_status. Si le statut de la commande est ROK, la valeur est `null`.

Accès uniquement sur `SMPP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`message:` ***Tampon***

Le contenu du champ short_message sur les messages DELIVER_SM et SUBMIT_SM. La valeur est `null` s'il n'est pas disponible ou n'est pas applicable.

Accès uniquement sur `SMPP_REQUEST` événements ; dans le cas contraire, une erreur se produira.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. Équivalent à `rspTimeToFirstByte - reqTimeToLastByte`. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `SMPP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `SMPP.commitRecord()` sur un `SMPP_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `clientZeroWnd`
- `command`
- `destination`
- `error`
- `receiverIsExternal`
- `reqSize`
- `reqTimeToLastByte`
- `rspSize`
- `rspTimeToFirstByte`
- `rspTimeToLastByte`
- `senderIsExternal`
- `serverIsExternal`
- `serverZeroWnd`
- `source`
- `processingTime`

`reqSize:` ***Numéro***

Le nombre d'octets de requête L7, à l'exclusion des en-têtes SMPP.

`reqTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la demande, exprimé en millisecondes. La valeur est `NaN` en cas de demandes mal formées ou abandonnées, ou si le délai n'est pas valide.

`rspSize:` ***Numéro***

Nombre d'octets de réponse L7, à l'exclusion des en-têtes SMPP.

Accès uniquement sur SMPP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspTimeToFirstByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le premier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur SMPP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur SMPP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`source:` ***Corde***

L'adresse source telle que spécifiée dans SMPP_REQUEST. La valeur est `null` si cela n'est pas disponible pour le type de commande actuel.

SMTP

Le SMTP la classe vous permet de stocker des métriques et d'accéder aux propriétés sur SMTP_REQUEST et SMTP_RESPONSE événements.

Évènements

SMTP_OPEN

Fonctionne sur chaque message d'accueil SMTP traité par l'équipement.

SMTP_REQUEST

S'exécute sur chaque demande SMTP traitée par l'équipement.

SMTP_RESPONSE

S'exécute sur chaque réponse SMTP traitée par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un SMTP_RESPONSE événement. Enregistrer les validations sur SMTP_REQUEST les événements ne sont pas pris en charge.

Pour consulter les propriétés par défaut attribuées à l' objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`dataSize:` ***Numéro***

Taille de la pièce jointe, exprimée en octets.

`domain:` ***Corde***

Le domaine de l'adresse d'où provient le message.

`error:` ***Corde***

Le code d'erreur correspondant au code d'état.

Accès uniquement sur SMTP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`headers:` ***Objet***

Objet qui permet d'accéder aux noms et aux valeurs des en-têtes SMTP.

La valeur du `headers` la propriété est la même lorsqu'on y accède sur l'un ou l'autre `SMTP_REQUEST` ou le `SMTP_RESPONSE` événement.

`isEncrypted:` ***Booléen***

La valeur est `true` si l'application est cryptée avec STARTTLS.

`isReqAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la demande SMTP ne soit terminée.

`isRspAborted:` ***Booléen***

La valeur est `true` si la connexion est fermée avant que la réponse SMTP ne soit terminée.

Accès uniquement sur `SMTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`method:` ***Corde***

La méthode SMTP.

`processingTime:` ***Numéro***

Le temps de traitement du serveur, exprimé en millisecondes. Équivalent à `rspTimeToFirstByte - reqTimeToLastByte`. La valeur est `NaN` en cas de réponses mal formées ou abandonnées ou si le timing n'est pas valide.

Accès uniquement sur `SMTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`recipientList:` ***Tableau de chaînes***

Liste des adresses des destinataires.

La valeur du `recipientList` la propriété est la même lorsqu'on y accède sur l'un ou l'autre `SMTP_REQUEST` ou le `SMTP_RESPONSE` événement.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `SMTP.commitRecord()` sur un `SMTP_RESPONSE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `clientZeroWnd`
- `dataSize`
- `domain`
- `error`
- `isEncrypted`
- `isReqAborted`
- `isRspAborted`
- `method`
- `processingTime`
- `receiverIsExternal`
- `recipient`
- `recipientList`
- `reqBytes`
- `reqL2Bytes`
- `reqPkts`
- `reqRTO`
- `reqSize`
- `reqTimeToLastByte`
- `roundTripTime`
- `rspBytes`

- rspL2Bytes
- rspPkts
- rspRTO
- rspSize
- rspTimeToFirstByte
- rspTimeToLastByte
- sender
- senderIsExternal
- serverIsExternal
- serverZeroWnd
- statusCode
- statusText

Accédez à l'objet d'enregistrement uniquement sur SMTP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`reqBytes:` ***Numéro***

Le nombre de L4 octets de demande, à l'exception des en-têtes L4.

`reqL2Bytes:` ***Numéro***

Le nombre de L2 octets de demande, y compris les en-têtes L2.

`reqPkts:` ***Numéro***

Le nombre de paquets de demandes.

`reqRTO:` ***Numéro***

Le numéro de demande délais de retransmission (RTO).

`reqSize:` ***Numéro***

Le nombre d'octets de requête L7, à l'exclusion des en-têtes SMTP.

`reqTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la demande, exprimé en millisecondes. La valeur est `NaN` en cas de demandes mal formées ou abandonnées, ou si le délai n'est pas valide.

`reqZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la demande.

`roundTripTime:` ***Numéro***

Le temps TCP aller-retour (RTT) médian, exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`rspBytes:` ***Numéro***

Le nombre de L4 octets de réponse, à l'exclusion de la surcharge du protocole L4, telle que les ACK, les en-têtes et les retransmissions.

`rspL2Bytes:` ***Numéro***

Le nombre de L2 octets de réponse, y compris les surcharges liées au protocole, telles que les en-têtes.

`rspPkts:` ***Numéro***

Le nombre de paquets de réponse.

`rspRTO:` ***Numéro***

Le nombre de réponses délais de retransmission (RTO).

`rspSize:` ***Numéro***

Nombre d'octets de réponse L7, à l'exclusion des en-têtes SMTP.

Accès uniquement sur SMTP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

`rspTimeToFirstByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le premier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur `SMTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspTimeToLastByte:` ***Numéro***

Temps écoulé entre le premier octet de la demande et le dernier octet de la réponse, exprimé en millisecondes. La valeur est `NaN` en cas de réponses mal formées ou abandonnées, ou si le timing n'est pas valide.

Accès uniquement sur `SMTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`rspZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles dans la réponse.

`sender:` ***Corde***

L'expéditeur du message.

`statusCode:` ***Numéro***

Le code d'état SMTP de la réponse ou du message d'accueil.

Accès uniquement sur `SMTP_RESPONSE` ou `SMTP_OPEN` événements ; dans le cas contraire, une erreur se produira.

`statusText:` ***Corde***

La réponse multiligne ou la chaîne de message d'accueil.

Accès uniquement sur `SMTP_RESPONSE` ou `SMTP_OPEN` événements ; dans le cas contraire, une erreur se produira.

SNMP

Le SNMP la classe vous permet de stocker des métriques et d'accéder à des propriétés sur `SNMP_REQUEST`, `SNMP_RESPONSE`, et `SNMP_MESSAGE` événements.

Évènements

`SNMP_REQUEST`

S'exécute sur chaque requête SNMP traitée par l'équipement.

`SNMP_RESPONSE`

S'exécute sur chaque réponse SNMP traitée par l'équipement.

`SNMP_MESSAGE`

S'exécute sur les messages SNMP qui ne respectent pas le comportement typique des requêtes et des réponses . Ni le `SNMP_REQUEST` événement ni le `SNMP_RESPONSE` l'événement s'exécute sur ces messages. Ces messages incluent des demandes envoyées par un serveur à un client et des réponses envoyées par un client à un serveur. Ces messages incluent également des interruptions SNMP, c'est-à-dire des messages envoyés par le serveur qui ne demandent pas de réponse.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `SNMP_REQUEST`, `SNMP_RESPONSE`, ou `SNMP_MESSAGE` événement. Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez la propriété d'enregistrement ci-dessous.

Si le `commitRecord()` méthode est appelée sur un `SNMP_REQUEST` événement, l'enregistrement n'est créé que lorsque `SNMP_RESPONSE` l'événement se déroule. Si le `commitRecord()` la

méthode est appelée à la fois sur SNMP_REQUEST et le correspondant SNMP_RESPONSE, un seul enregistrement est créé pour la demande et la réponse, même si commitRecord() La méthode est appelée plusieurs fois lors des mêmes événements déclencheurs.

Propriétés

error: ***Corde***

Message d'erreur SNMP.

community: ***Corde***

La chaîne de communauté SNMP.

payload: ***Tampon***

L'objet Buffer qui contient les octets de charge utile bruts de la transaction d'événement. La mémoire tampon contient les 1 024 premiers octets de la charge utile.

pduType: ***Corde***

Type d'unité de données de protocole (PDU).

record: ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à SNMP.commitRecord () sur SNMP_REQUEST, SNMP_RESPONSE, ou SNMP_MESSAGE événement.

L'événement pour lequel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

SNMP_REQUEST	SNMP_RESPONSE	SNMP_MESSAGE
client	client	community
clientAddr	clientAddr	error
clientIsExternal	clientIsExternal	flowId
clientPort	clientPort	pduType
community	community	receiver
flowId	error	receiverAddr
pduType	flowId	receiverPort
server	pduType	receiverIsExternal
serverAddr	server	sender
serverIsExternal	serverAddr	senderAddr
serverPort	serverIsExternal	senderPort
version	serverPort	senderIsExternal
vlan	version	version
	vlan	vlan

version: ***Corde***

Version du protocole SNMP.

SOCKS

La classe Socket Secure (SOCKS) vous permet de stocker des métriques et d'accéder à des propriétés sur SOCKS_REQUEST et SOCKS_RESPONSE événements.

Évènements

SOCKS_REQUEST

S'exécute sur tous les messages SOCKS traités par l'équipement.

SOCKS_RESPONSE

S'exécute sur tous les messages SOCKS traités par l'équipement.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur SOCKS_RESPONSE événement. Enregistrez les validations le SOCKS_REQUEST les événements ne sont pas pris en charge. Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez la propriété d'enregistrement ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu' une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`authResult:` ***Numéro***

Indique si l'authentification a réussi. Les valeurs suivantes sont valides.

Valeur	Descriptif
0	A réussi
1	Échoué

Note: Si le protocole est SOCKS4, la valeur est toujours `0` car SOCKS4 ne prend pas en charge l'authentification.

`authType:` ***Numéro***

La méthode dqu`authentification qui a été négociée entre le serveur et le client.

`command:` ***Numéro***

Code numérique de la commande SOCKS demandée par le client. Les codes de commande suivants sont valides.

Code	Descriptif
1	Connecter un flux TCP
2	Liez le port TCP
3	Port UDP associé

`requestAddress:` ***Adresse IP***

Le IPAddress objet pour l'adresse spécifiée par le client dans la demande.

`requestPort:` ***Numéro***

Numéro de port spécifié par le client dans la demande.

`responseAddress:` ***Adresse IP***

Le IPAddress objet pour l'adresse spécifiée par le serveur dans la réponse.

`responsePort:` ***Numéro***

Numéro de port spécifié par le serveur dans la réponse.

`result:` ***Numéro***

Le code dquo spécifié par le serveur dans la réponse.

`username:` ***Corde***

Le nom de l'utilisateur spécifié par le client pour l'authentification.

`version:` ***Numéro***

La version du protocole SOCKS.

SSH

Secure Socket Shell (SSH) est un réseau protocole qui fournit une méthode sécurisée pour la connexion à distance et d'autres services réseau sur un réseau non sécurisé. Le SSH un objet de classe vous permet de stocker des métriques et d'accéder aux propriétés sur SSH_CLOSE, SSH_OPEN et SSH_TICK événements.

Évènements

`SSH_CLOSE`

S'exécute lorsque la connexion SSH est interrompue en raison de sa fermeture, de son expiration ou de son abandon.

`SSH_OPEN`

S'exécute lorsque la connexion SSH est complètement établie pour la première fois après avoir négocié les informations de session . Si la négociation échoue parce que l'échange de clés n'est pas valide, SSH_OPEN l'événement s'exécute lorsqu'il y a un échange non valide, puis le SSH_TICK et SSH_CLOSE les événements se succèdent immédiatement.

Si une connexion se ferme avant SSH_OPEN court, SSH_OPEN, SSH_TICK, et SSH_CLOSE s'exécutent en succession immédiate.

`SSH_TICK`

S'exécute périodiquement sur les flux SSH.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un SSH_OPEN, SSH_CLOSE, ou SSH_TICK événement.

L'événement détermine les propriétés qui sont validées dans l'objet d'enregistrement. Pour consulter les propriétés validées pour chaque événement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `.commitRecord` est appelé plusieurs fois pour le même enregistrement unique.

Propriétés

`clientBytes:` ***Numéro***

Sur un SSH_CLOSE événement, le nombre incrémentiel de niveaux d'application client octets observés depuis le dernier SSH_TICK événement. Ne précise pas le nombre total d'octets pour la session.

`clientCipherAlgorithm:` ***Corde***

L'algorithme de chiffrement sur le SSH client.

`clientCompressionAlgorithm:` ***Corde***

Algorithme de compression appliqué aux données transférées via la connexion par le client SSH .

`clientCompressionAlgorithmsClientToServer:` ***Corde***

Algorithmes de compression pris en charge par le client SSH pour les communications client-serveur .

`clientCompressionAlgorithmsServerToClient:` ***Corde***

Algorithmes de compression pris en charge par le client SSH pour les communications entre serveurs et clients .

`clientEncryptionAlgorithmsClientToServer:` ***Corde***

Algorithmes de chiffrement pris en charge par le client SSH pour les communications client-serveur .

`clientEncryptionAlgorithmsServerToClient:` ***Corde***

Algorithmes de chiffrement pris en charge par le client SSH pour les communications entre serveurs et clients .

`clientImplementation:` ***Corde***

L'implémentation SSH installée sur le client, telle qu'OpenSSH ou PUTTY.

`clientKexAlgorithms:` ***Corde***

Les algorithmes d'échange de clés SSH pris en charge par le client.

`clientL2Bytes:` ***Numéro***

Le nombre incrémentiel de L2 octets du client observés depuis le dernier `SSH_TICK` événement. Ne précise pas le nombre total d'octets pour la session.

Accès uniquement sur `SSH_CLOSE` et `SSH_TICK` événements ; dans le cas contraire, une erreur se produira.

`clientMacAlgorithm:` ***Corde***

L'algorithme MAC (Method Authentication Code) sur le client SSH.

`clientMacAlgorithmsClientToServer:` ***Corde***

Algorithmes MAC (Method Authentication Code) pris en charge par le client SSH pour les communications client-serveur.

`clientMacAlgorithmsServerToClient:` ***Corde***

Algorithmes MAC (Method Authentication Code) pris en charge par le client SSH pour les communications entre serveurs et clients.

`clientPkts:` ***Numéro***

Le nombre incrémentiel de paquets clients observés depuis le dernier `SSH_TICK` événement. Ne précise pas le nombre total de paquets pour la session.

Accès uniquement sur `SSH_CLOSE` et `SSH_TICK` événements ; dans le cas contraire, une erreur se produira.

`clientRTO:` ***Numéro***

Le nombre croissant de clients délais de retransmission (RTO) observés depuis le dernier `SSH_TICK` événement. Ne précise pas le nombre total de RTO pour la session.

Accès uniquement sur `SSH_CLOSE` et `SSH_TICK` événements ; sinon, une erreur se produira.

`clientVersion:` ***Corde***

La version de SSH sur le client.

`clientZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par le client.

Accès uniquement sur `SSH_OPEN`, `SSH_CLOSE`, ou `SSH_TICK` événements ; sinon, une erreur se produira.

`duration:` ***Numéro***

Durée, exprimée en millisecondes, de la connexion SSH.

Accès uniquement sur `SSH_CLOSE` événements ; sinon, une erreur se produira.

`hasshAlgorithms:` ***Corde***

Chaîne contenant les algorithmes d'échange de clés SSH, de chiffrement, d'authentification des messages et de compression pris en charge par le client pour les communications SSH. Ces algorithmes sont communiqués dans le paquet SSH_MSG_KEXINIT envoyé au début d'une connexion SSH.

`hassh:` ***Corde***

Un hachage MD5 de la chaîne HashAlgorithms.

`hasshServerAlgorithms:` ***Corde***

Chaîne contenant les algorithmes d'échange de clés SSH, de chiffrement, d'authentification des messages et de compression pris en charge par le serveur pour les communications SSH. Ces algorithmes sont communiqués dans le paquet SSH_MSG_KEXINIT envoyé au début d'une connexion SSH.

`hasshServer:` ***Corde***

Un hachage MD5 de la chaîne HashServerAlgorithms.

`kexAlgorithm:` ***Corde***

L'algorithme d'échange de clés (Kex) sur la connexion SSH.

`messageNumbers:` ***Tableau de nombres***

Les identifiants numériques des messages SSH échangés, listés par ordre chronologique. Le tableau ne peut pas contenir plus de 50 entrées. Si plus de 50 messages sont échangés, le tableau contient les 50 identifiants les plus récents.

Accès uniquement sur `SSH_OPEN` événements ; dans le cas contraire, une erreur se produira.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `SSH.commitRecord()` sur l'un ou l'autre `SSH_OPEN`, `SSH_CLOSE`, ou `SSH_TICK` événement.

L'événement au cours duquel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

SSH_TICK	SSH_OPEN	SSH_CLOSE
clientCipherAlgorithm	clientCipherAlgorithm	clientCipherAlgorithm
clientCompressionAlgorithm	clientCompressionAlgorithm	clientCompressionAlgorithm
clientImplementation	clientImplementation	clientImplementation
clientIsExternal	clientIsExternal	clientIsExternal
clientMacAlgorithm	clientMacAlgorithm	clientMacAlgorithm
clientVersion	clientVersion	clientVersion
clientZeroWnd	clientZeroWnd	clientZeroWnd
kexAlgorithm	kexAlgorithm	kexAlgorithm
receiverIsExternal	receiverIsExternal	receiverIsExternal
senderIsExternal	senderIsExternal	senderIsExternal
serverCipherAlgorithm	serverCipherAlgorithm	serverCipherAlgorithm

SSH_TICK	SSH_OPEN	SSH_CLOSE
serverCompressionAlgorithm	serverCompressionAlgorithm	serverCompressionAlgorithm
serverImplementation	serverImplementation	serverImplementation
serverIsExternal	serverIsExternal	serverIsExternal
serverMacAlgorithm	serverMacAlgorithm	serverMacAlgorithm
serverVersion	serverVersion	serverVersion
serverZeroWnd	serverZeroWnd	serverZeroWnd
		duration

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`serverBytes:` ***Numéro***

Le nombre incrémentiel d'octets de serveur au niveau de l'application observés depuis le dernier `SSH_TICK` événement. Ne précise pas le nombre total d'octets pour la session.

Accès uniquement sur `SSH_CLOSE` et `SSH_TICK` événements ; dans le cas contraire, une erreur se produira.

`serverCipherAlgorithm:` ***Corde***

L'algorithme de chiffrement sur le serveur SSH.

`serverCompressionAlgorithm:` ***Corde***

Renvoie le type de compression appliqué aux données transférées via la connexion par le serveur SSH.

`serverCompressionAlgorithmsClientToServer:` ***Corde***

Algorithmes de compression pris en charge par le serveur SSH pour les communications client-serveur .

`serverCompressionAlgorithmsServerToClient:` ***Corde***

Algorithmes de compression pris en charge par le serveur SSH pour les communications entre le serveur et le client .

`serverEncryptionAlgorithmsClientToServer:` ***Corde***

Algorithmes de chiffrement pris en charge par le serveur SSH pour les communications client-serveur .

`serverEncryptionAlgorithmsServerToClient:` ***Corde***

Algorithmes de chiffrement pris en charge par le serveur SSH pour les communications entre le serveur et le client .

`serverHostKey:` ***Corde***

Le codage base64 de la clé SSH publique envoyée par le serveur au client.

`serverHostKeyType:` ***Corde***

Type de clé SSH publique envoyée par le serveur au client, telle que ssh-rsa ou ssh-ed25519.

`serverImplementation:` ***Corde***

L'implémentation SSH installée sur le serveur, telle qu'OpenSSH ou PUTTY.

`serverKexAlgorithms:` ***Corde***

Les algorithmes d'échange de clés SSH pris en charge par le serveur.

`serverL2Bytes:` ***Numéro***

Le nombre incrémentiel de L2 octets de serveur observés depuis le dernier `SSH_TICK` événement. Ne précise pas le nombre total d'octets pour la session.

Accès uniquement sur SSH_CLOSE et SSH_TICK événements ; dans le cas contraire, une erreur se produira.

`serverMacAlgorithm:` ***Corde***

L'algorithme MAC (Method Authentication Code) sur le serveur SSH.

`serverMacAlgorithmsClientToServer:` ***Corde***

Algorithmes MAC (Method Authentication Code) pris en charge par le serveur SSH pour les communications client-serveur.

`serverMacAlgorithmsServerToClient:` ***Corde***

Algorithmes MAC (Method Authentication Code) pris en charge par le serveur SSH pour les communications entre serveur et client.

`serverPkts:` ***Numéro***

Le nombre incrémentiel de paquets de serveur observés depuis le dernier SSH_TICK événement. Ne précise pas le nombre total de paquets pour la session.

Accès uniquement sur SSH_CLOSE et SSH_TICK événements ; dans le cas contraire, une erreur se produira.

`serverRTO:` ***Numéro***

Le nombre incrémentiel de serveurs délais de retransmission (RTO) observés depuis le dernier SSH_TICK événement. Ne précise pas le nombre total de RTO pour la session.

Accès uniquement sur SSH_CLOSE et SSH_TICK événements ; sinon, une erreur se produira.

`serverVersion:` ***Corde***

Version de SSH sur le serveur.

`serverZeroWnd:` ***Numéro***

Nombre de fenêtres nulles envoyées par le serveur.

Accès uniquement sur SSH_OPEN, SSH_CLOSE, ou SSH_TICK événements ; sinon, une erreur se produira.

SSL

Le SSL la classe vous permet de stocker des métriques et d'accéder aux propriétés sur SSL_OPEN, SSL_CLOSE, SSL_ALERT, SSL_RECORD, SSL_HEARTBEAT, et SSL_RENEGOTIATE événements.

Évènements

SSL_ALERT

S'exécute lorsqu'un enregistrement d'alerte TLS est échangé.

SSL_CLOSE

S'exécute lorsque la connexion TLS est arrêtée.

SSL_HEARTBEAT

S'exécute lorsqu'un enregistrement de pulsation TLS est échangé.

SSL_OPEN

S'exécute lorsque la connexion TLS est établie pour la première fois.

SSL_PAYLOAD

S'exécute lorsque la charge utile TLS déchiffrée correspond aux critères configurés dans le déclencheur associé.

Selon le flux, la charge utile se trouve dans les propriétés suivantes :

- `Flow.payload1`
- `Flow.payload2`

- `Flow.client.payload`
- `Flow.server.payload`
- `Flow.sender.payload`
- `Flow.receiver.payload`

Des options de charge utile supplémentaires sont disponibles lorsque vous créez un déclencheur qui s'exécute sur cet événement. Voir Options de déclencheur avancées pour plus d' informations.

`SSL_RECORD`

S'exécute lorsqu'un enregistrement TLS est échangé.

`SSL_RENEGOTIATE`

S'exécute lors de la renégociation TLS.

Méthodes

`addApplication(name:` ***Corde*** `):` ***vide***

Associe une session TLS à l'application nommée pour collecter les données métriques TLS relatives à la session. Par exemple, vous pouvez appeler `SSL.addApplication()` pour associer les données de certificat TLS à une application.

Une fois qu'une session TLS est associée à une application, ce couplage est permanent pendant toute la durée de vie de la session.

Appelez uniquement au `SSL_OPEN` événements ; sinon, une erreur se produira.

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré uniquement le `SSL_ALERT`, `SSL_CLOSE`, `SSL_HEARTBEAT`, `SSL_OPEN`, ou `SSL_RENEGOTIATE` événements. Enregistrez les engagements le `SSL_PAYLOAD` et `SSL_RECORD` les événements ne sont pas pris en charge.

Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` La méthode est appelée plusieurs fois pour le même enregistrement unique.

`getClientExtensionData(extension_name | extension_id):` ***Tampon | Nul***

Renvoie les données de l'extension spécifiée si l'extension a été transmise dans le cadre du message Hello du client. Retours `null` si le message ne contient pas de données.

Appelez uniquement au `SSL_OPEN` et `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`getServerExtensionData(extension_name | extension_id):` ***Tampon | Nul***

Renvoie les données pour l'extension spécifiée si l'extension a été transmise dans le cadre du `Hello` message du serveur. Retours `null` si le message ne contient pas de données.

Appelez uniquement au `SSL_OPEN` et `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`hasClientExtension(extension_name | extension_id):` ***booléen***

Retours `true` pour l'extension spécifiée si l'extension a été transmise dans le cadre du `Hello` message du client.

Appelez uniquement au `SSL_OPEN` et `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`hasServerExtension(extension_name | extension_id):` ***booléen***

Retours `true` pour l'extension spécifiée si l'extension a été transmise dans le cadre du `Hello` message du serveur.

Appelez uniquement au SSL_OPEN et SSL_RENEGOTIATE événements ; sinon, une erreur se produira.

Le tableau suivant fournit la liste des extensions TLS connues.

IDENTIFIANT	Nom
0	server_name
1	max_fragment_length
2	client_certificate_url
3	trusted_ca_keys
4	truncated_hmac
5	status_request
6	user_mapping
7	client_authz
8	server_authz
9	cert_type
10	supported_groups
11	ec_point_formats
12	srp
13	signature_algorithms
14	use_srtp
15	heartbeat
16	application_layer_protocol_negotiation
17	status_request_v2
18	signed_certificate_timestamp
19	client_certificate_type
20	server_certificate_type
27	compress_certificate
28	record_size_limit
29	pwd_protect
30	pwd_clear
31	password_salt
35	session_ticket
41	pre_shared_key
42	early_data
43	supported_versions
44	cookie
45	psk_key_exchange_modes

IDENTIFIANT	Nom
47	certificate_authorities
48	oid_filters
49	post_handshake_auth
50	signature_algorithms_cert
51	key_share
65281	renegotiation_info
65486	encrypted_server_name

Les extensions suivantes sont envoyées par les applications pour tester si les serveurs peuvent gérer des extensions inconnues. Pour plus d'informations sur ces extensions, voir Appliquer GREASE à l'extensibilité TLS.

- 2570
- 6682
- 10794
- 14906
- 19018
- 23130
- 27242
- 31354
- 35466
- 39578
- 43690
- 47802
- 51914
- 56026
- 60138
- 64250

Propriétés

`alertCode`: ***Numéro***

Représentation numérique de l'alerte TLS. Le tableau suivant affiche les alertes TLS possibles, qui sont définies dans le `AlertDescription` structure de données dans la RFC 2246 :

Alerte	Numéro
close_notify	0
unexpected_message	10
bad_record_mac	20
decryption_failed	21
record_overflow	22
decompression_failure	30
handshake_failure	40
bad_certificate	42

Alerte	Numéro
unsupported_certificate	43
certificate_revoked	44
certificate_expired	45
certificate_unknown	46
illegal_parameter	47
unknown_ca	48
access_denied	49
decode_error	50
decrypt_error	51
export_restriction	60
protocol_version	70
insufficient_security	71
internal_error	80
user_canceled	90
no_renegotiation	100

Si la session est opaque, la valeur est `SSL.ALERT_CODE_UNKNOWN (null)`.

Accès uniquement sur SSL_ALERT événements ; sinon, une erreur se produira.

`alertCodeName:` ***Corde***

Nom de l'alerte TLS associée au code d'alerte. Consultez les `alertCode` propriété pour les noms d'alerte associés aux codes d'alerte. La valeur est `null` si aucun nom n'est disponible pour le code d'alerte associé.

Accès uniquement sur SSL_ALERT événements ; sinon, une erreur se produira.

`alertLevel:` ***Numéro***

Représentation numérique du niveau d'alerte TLS. Les niveaux d'alerte possibles suivants sont définis dans le `AlertLevel` structure de données dans la RFC 2246 :

- `warning (1)`
- `fatal (2)`

Si la session est opaque, la valeur est `SSL.ALERT_LEVEL_UNKNOWN (null)`.

Accès uniquement sur SSL_ALERT événements ; sinon, une erreur se produira.

`certificate:` ***Certificat SSL***

L'objet de certificat de serveur TLS associé à la communication. Chaque objet contient les propriétés suivantes :

`authorityInfoAccess:` ***Objet***

Objet contenant des informations provenant de l'extension Authority Information Access, qui spécifie des informations sur l'autorité de certification (CA). L'objet contient les champs suivants :

`location:` ***Corde***

URL du répondeur OCSP (Online Certificate Status Protocol) qui permet de vérifier si le certificat est valide.

`method:` ***Corde***

L'OID de la méthode permettant d'accéder à l'émetteur du certificat.

`authorityKeyIdentifier:` ***Corde | Null***

L'identifiant de la clé publique de l'autorité de certification (CA), exprimé sous forme de chaîne d'octets.

Note: Ce champ ne contient ni l' émetteur ni le numéro de série de la certification de l'autorité.

`basicConstraints:` ***Objet***

Objet contenant des informations provenant de l' extension Basic Constraints, qui spécifie le type de sujet du certificat. L' objet contient les champs suivants :

`ca:` ***Booléen***

Indique si l'objet du certificat est une autorité de certification.

`pathlen:` ***Numéro***

Le nombre maximum de certificats pouvant apparaître dans la chaîne de certificats après ce certificat.

`certificatePolicies:` ***Tableau de cordes***

Tableau d'OID pour les politiques spécifiées dans l'extension Certificate Policies. Les qualificatifs ne sont pas inclus dans ce tableau.

`crlDistributionPoints:` ***Tableau de cordes***

Tableau d'objets contenant des informations sur les serveurs hébergeant des listes de révocation de certificats (CRL) pour le certificat de serveur. Les serveurs sont spécifiés dans l'extension du point de distribution CRL (CDP). Chaque objet contient les champs suivants :

Émetteur CRL : Tableau de cordes

Un ensemble d'emplacements où le certificat de l'émetteur de la CRL peut être récupéré.

`distPoint:` ***Tableau de cordes***

Un ensemble d'emplacements où la CRL peut être récupérée.

`reasons:` ***Tableau de cordes***

Tableau de codes de motif indiquant les raisons pour lesquelles le certificat pourrait être révoqué par le point de distribution CRL.

`extensionOIDs:` ***Tableau de cordes***

Tableau d'OID pour les extensions X509 spécifiées dans le certificat.

`extendedKeyUsage:` ***Tableau de cordes***

Tableau des utilisations de la clé publique du certificat de serveur spécifié dans l'extension Extended Key Usage. Le tableau peut contenir les chaînes suivantes :

- `serverAuth`
- `clientAuth`
- `emailProtection`
- `codeSigning`
- `OCSPSigning`
- `timeStamping`
- `anyExtendedKeyUsage`
- `nsSGC`

`fingerprint:` ***Corde***

Représentation hexadécimale du hachage SHA-1 du certificat. La chaîne ne contient aucun délimiteur, comme illustré dans l'exemple suivant :

```
55F30E6D49E19145CF680E8B7E3DC8FC7041DC81
```

Le hachage du certificat SHA-1 apparaît dans la boîte de dialogue du certificat de serveur de la plupart des navigateurs.

`fingerprintSHA256:` ***Corde***

Représentation hexadécimale du hachage SHA-256 du certificat. La chaîne ne contient aucun délimiteur, comme illustré dans l'exemple suivant :

```
468C6C84DB844821C9CCB0983C78D1CC05327119B894B5CA1C6A1318784D3675
```

Le hachage du certificat SHA-256 apparaît dans la boîte de dialogue du certificat de serveur de la plupart des navigateurs.

`getExtensionDataByOID(extension_oid):` ***Tampon***

Méthode qui renvoie un objet tampon contenant la valeur de l'extension spécifiée, exprimée sous forme de chaîne d'octets. Renvoie la valeur null si l'OID n'existe pas ou si le certificat du serveur ne contient pas l'extension.

`inhibitAnyPolicy:` ***Numéro***

Le nombre spécifié dans l'extension Inhibit AnyPolicy, qui limite le nombre de certificats auxquels l'extension AnyPolicy est appliquée. Le nombre indique combien de certificats supplémentaires non auto-émis de la chaîne sont affectés par l'extension AnyPolicy.

`isSelfSigned:` ***Booléen***

La valeur est `true` si le certificat du serveur est auto-signé.

`issuer:` ***Corde***

Nom commun de l'émetteur du certificat de serveur. La valeur est `null` si l'émetteur n'est pas disponible.

`issuerAlternativeNames:` ***Tableau de cordes***

Tableau de noms alternatifs d'émetteur (IANS) spécifiés dans le certificat du serveur.

`issuerDistinguishedName:` ***Objet***

Objet contenant des informations sur le nom distinctif de l'émetteur du certificat. Chaque objet contient les propriétés suivantes :

`commonName:` ***Corde***

Le nom commun (CN).

`country:` ***Tableau de cordes***

Le nom du pays (C).

`emailAddress:` ***Corde***

L'adresse e-mail.

`organization:` ***Tableau de cordes***

Le nom de l'organisation (O).

`organizationalUnit:` ***Tableau de cordes***

Le nom de l'unité organisationnelle (OU).

`locality:` ***Tableau de cordes***

Le nom de la localité (L).

`stateOrProvince:` ***Tableau de cordes***

Le nom de l'État ou de la province (ST).

`keySize:` ***Numéro***

Taille de clé du certificat de serveur.

`keyUsage:` ***Tableau de cordes***

Tableau des utilisations de la clé publique du certificat de serveur spécifié dans l'extension Key Usage. Le tableau peut contenir les chaînes suivantes :

- `digitalSignature`
- `nonRepudiation`
- `keyEncipherment`
- `dataEncipherment`
- `keyAgreement`
- `keyCertSign`
- `cRLSign`
- `encipherOnly`
- `decipherOnly`

`notAfter:` ***Numéro***

Heure d'expiration du certificat de serveur, exprimée en UTC.

`notBefore:` ***Numéro***

Heure de début du certificat de serveur, exprimée en UTC. Le certificat de serveur n'est pas valide avant cette date.

`nsComment:` ***Corde***

Le commentaire spécifié dans l'extension Netscape Comment. Ce commentaire est parfois affiché dans les navigateurs lorsque les utilisateurs consultent le certificat du serveur.

`ocspNoCheck:` ***Booléen***

Indique si le certificat de signature peut être approuvé sans vérification par le répondeur OCSP.

`payload:` ***Tampon***

Le Tampon objet qui contient les octets de charge utile bruts du certificat de serveur.

`policyConstraints:` ***Objet***

Objet contenant des informations provenant de l' extension Policy Constraints, qui spécifie les contraintes de validation pour les certificats CA.

`requireExplicitPolicy:` ***Numéro***

Spécifie le nombre maximum de certificats adjacents dans la chaîne qui n'ont pas besoin de spécifier de politique explicite.

`inhibitPolicyMapping:` ***Numéro***

Spécifie le nombre maximum de certificats adjacents dans la chaîne de certificats avant que les mappages de politiques ne soient ignorés.

`policyMappings:` ***Tableau d' objets***

Tableau d'objets contenant des informations provenant de l'extension Policy Mappings, qui indique des politiques équivalentes les unes aux autres. Chaque objet contient les champs suivants.

`issuerDomainPolicy:` ***Corde***

L'OID de la politique de l'émetteur.

`subjectDomainPolicy:` ***Corde***

L'OID de la politique en question.

`publicKeyCurveName:` ***Corde***

Nom de la courbe elliptique standard sur laquelle est basée la cryptographie de la clé publique. Cette valeur est déterminée par l'OID ou les paramètres de courbe explicites spécifiés dans le certificat.

`publicKeyExponent:` ***Corde | Null***

Une représentation hexadécimale sous forme de chaîne de l'exposant de clé publique. La chaîne est affichée dans la boîte de dialogue du certificat client de la plupart des navigateurs, mais sans espaces.

`publicKeyHasExplicitCurve:` ***Booléen | Null***

Indique si le certificat spécifie des paramètres explicites pour la courbe elliptique de la clé publique.

`publicKeyModulus:` ***Corde | Null***

Une représentation hexadécimale sous forme de chaîne du module de clé publique. La chaîne est affichée dans la boîte de dialogue du certificat client de la plupart des navigateurs, mais sans espace, comme `010001`

`serial:` ***Corde | Null***

Le numéro de série attribué au certificat par l' autorité de certification (CA).

`signatureAlgorithm:` ***Corde | Null***

Algorithme appliqué pour signer le certificat du serveur. Le tableau suivant présente certaines des valeurs possibles :

RFC	Algorithme
`RFC 3279`	• `md2WithRSAEncryption` • `md5WithRSAEncryption` • `sha1WithRSAEncryption`
`RFC 4055`	• `sha224WithRSAEncryption` • `sha256WithRSAEncryption \` • `sha384WithRSAEncryption` • `sha512WithRSAEncryption`
`RFC 4491`	• `id-GostR3411-94-with-Gost3410-94` • `id-GostR3411-94-with-Gost3410-2001`

`subject:` ***Corde***

Le nom commun du sujet (CN) du certificat de serveur.

`subjectAlternativeNames:` ***Array***

Tableau de chaînes correspondant aux noms alternatifs de sujet (SAN) inclus dans le certificat de serveur. Les SAN pris en charge sont les noms DNS, les adresses e-mail, les URI et les adresses IP.

`subjectDistinguishedName:` ***Objet***

Objet contenant des informations sur le nom distinctif du sujet du certificat. Chaque objet contient les propriétés suivantes :

`commonName:` ***Corde***

Le nom commun (CN).

`country:` ***Tableau de cordes***

Le nom du pays (C).

`emailAddress:` ***Corde***

L'adresse e-mail.

`organization:` ***Tableau de cordes***

Le nom de l'organisation (O).

`organizationalUnit:` ***Tableau de cordes***

Le nom de l'unité organisationnelle (OU).

`locality:` ***Tableau de cordes***

Le nom de la localité (L).

`stateOrProvince:` ***Tableau de cordes***

Le nom de l'État ou de la province (ST).

`subjectKeyIdentifier:` ***Corde***

Identifiant de la clé publique du sujet du certificat, exprimé sous forme de chaîne d'octets.

`certificates:` ***Tableau d'objets***

Tableau d'objets de certificat pour chaque certificat TLS intermédiaire. Le certificat d'entité finale, également appelé certificat feuille, est le premier objet du tableau ; cet objet est également renvoyé par le `certificate` propriété.

`cipherSuite:` ***Corde***

Chaîne représentant la suite de chiffrement cryptographique négociée entre le serveur et le client.

`cipherSuitesHex:` ***Corde***

Représentation hexadécimale de la suite de chiffrement cryptographique négociée entre le serveur et le client.

`cipherSuitesSupported:` ***Tableau d'objets | Nul***

Tableau d'objets avec les propriétés suivantes qui spécifient les suites de chiffrement prises en charge par le client TLS :

`name:` ***Corde***

Nom de la suite de chiffrement.

`type:` ***Numéro***

Le numéro de suite de chiffrement.

Accès uniquement sur `SSL_OPEN` ou `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`cipherSuiteType:` ***Numéro***

Valeur numérique qui représente la suite de chiffrement cryptographique négociée entre le serveur et le client. Les valeurs possibles sont définies par le registre de la suite de chiffrement TLS de l'IANA.

`clientBytes:` ***Numéro***

Le nombre d'octets envoyés par le client depuis la dernière `SSL_RECORD` événement.

Accès uniquement sur `SSL_RECORD` ou `SSL_CLOSE` événements ; sinon, une erreur se produira.

`clientCertificate:` ***Certificat SSL***

L'objet de certificat client TLS associé à la communication. Chaque objet contient les propriétés suivantes :

`authorityInfoAccess:` ***Objet***

Objet contenant des informations provenant de l'extension Authority Information Access, qui spécifie des informations sur l'autorité de certification (CA). L'objet contient les champs suivants :

`location:` ***Corde***

URL du répondeur OCSP (Online Certificate Status Protocol) qui permet de vérifier si le certificat est valide.

`method:` ***Corde***

L'OID de la méthode permettant d'accéder à l'émetteur du certificat.

`authorityKeyIdentifier:` ***Corde | Null***

L'identifiant de la clé publique de l'autorité de certification (CA), exprimé sous forme de chaîne d'octets.

Note: Ce champ ne contient ni l' émetteur ni le numéro de série de la certification de l'autorité.

`basicConstraints:` ***Objet***

Objet contenant des informations provenant de l' extension Basic Constraints, qui spécifie le type de sujet du certificat. L' objet contient les champs suivants :

`ca:` ***Booléen***

Indique si l'objet du certificat est une autorité de certification.

`pathlen:` ***Numéro***

Le nombre maximum de certificats pouvant apparaître dans la chaîne de certificats après ce certificat.

`certificatePolicies:` ***Tableau de cordes***

Tableau d'OID pour les politiques spécifiées dans l'extension Certificate Policies. Les qualificatifs ne sont pas inclus dans ce tableau.

`crlDistributionPoints:` ***Tableau de cordes***

Tableau d'objets contenant des informations sur les serveurs hébergeant des listes de révocation de certificats (CRL) pour le certificat client . Les serveurs sont spécifiés dans l'extension du point de distribution CRL (CDP). Chaque objet contient les champs suivants :

Émetteur CRL : Tableau de cordes

Un ensemble d'emplacements où le certificat de l'émetteur de la CRL peut être récupéré.

`distPoint:` ***Tableau de cordes***

Un ensemble d'emplacements où la CRL peut être récupérée.

`reasons:` ***Tableau de cordes***

Tableau de codes de motif indiquant les raisons pour lesquelles le certificat pourrait être révoqué par le point de distribution CRL.

`extensionOIDs:` ***Tableau de cordes***

Tableau d'OID pour les extensions X509 spécifiées dans le certificat client.

`extendedKeyUsage:` ***Tableau de cordes***

Tableau des utilisations de la clé publique du certificat client spécifié dans l'extension Extended Key Usage. Le tableau peut contenir les chaînes suivantes :

- `serverAuth`
- `clientAuth`
- `emailProtection`
- `codeSigning`
- `OCSPSigning`
- `timeStamping`
- `anyExtendedKeyUsage`
- `nsSGC`

`fingerprint:` ***Corde***

Représentation hexadécimale du hachage SHA-1 du certificat client. La chaîne ne contient aucun délimiteur, comme illustré dans l'exemple suivant :

```
55F30E6D49E19145CF680E8B7E3DC8FC7041DC81
```

`fingerprintSHA256:` ***Corde***

Représentation hexadécimale du hachage SHA-256 du certificat client. La chaîne ne contient aucun délimiteur, comme illustré dans l'exemple suivant :

```
468C6C84DB844821C9CCB0983C78D1CC05327119B894B5CA1C6A1318784D3675
```

`getExtensionDataByOID(extension_oid):` ***Tampon***

Méthode qui renvoie un objet tampon contenant la valeur de l'extension spécifiée, exprimée sous forme de chaîne d'octets. Renvoie la valeur null si l'OID n'existe pas ou si le certificat client ne contient pas l'extension.

`keySize:` ***Numéro***

Taille de clé du certificat client.

`keyUsage:` ***Tableau de cordes***

Tableau des utilisations de la clé publique du certificat client spécifié dans l'extension Key Usage. Le tableau peut contenir les chaînes suivantes :

- `digitalSignature`
- `nonRepudiation`
- `keyEncipherment`
- `dataEncipherment`
- `keyAgreement`
- `keyCertSign`
- `cRLSign`
- `encipherOnly`
- `decipherOnly`

`inhibitAnyPolicy:` ***Numéro***

Le nombre spécifié dans l'extension Inhibit AnyPolicy, qui limite le nombre de certificats auxquels l'extension AnyPolicy est appliquée. Le nombre indique combien de certificats supplémentaires non auto-émis de la chaîne sont affectés par l'extension AnyPolicy.

`isSelfSigned:` ***Booléen***

La valeur est `true` si le certificat client est auto-signé.

`issuer:` ***Corde | Null***

Nom commun de l'émetteur du certificat client. La valeur est `null` si l'émetteur n'est pas disponible.

`issuerDistinguishedName:` ***Objet***

Objet contenant des informations sur le nom distinctif de l'émetteur du certificat. Chaque objet contient les propriétés suivantes :

`commonName:` ***Corde***

Le nom commun (CN).

`country:` ***Tableau de cordes***

Le nom du pays (C).

`emailAddress:` ***Corde***

L'adresse e-mail.

`organization:` ***Tableau de cordes***

Le nom de l'organisation (O).

`organizationalUnit:` ***Tableau de cordes***

Le nom de l'unité organisationnelle (OU).

`locality:` ***Tableau de cordes***

Le nom de la localité (L).

`stateOrProvince:` ***Tableau de cordes***

Le nom de l'État ou de la province (ST).

`issuerAlternativeNames:` ***Tableau de cordes***

Tableau de noms alternatifs de l'émetteur (IANS) spécifiés dans le certificat client.

`notAfter:` ***Numéro***

L'heure d'expiration du certificat client, exprimée en UTC.

`notBefore:` ***Numéro***

Heure de début du certificat client, exprimée en UTC. Le certificat client n'est pas valide avant cette date.

`nsComment:` ***Corde***

Le commentaire spécifié dans l'extension Netscape Comment. Ce commentaire est parfois affiché dans les navigateurs lorsque les utilisateurs consultent le certificat client.

`ocspNoCheck:` ***Booléen***

Indique si le certificat de signature peut être approuvé sans vérification par le répondeur OCSP.

`payload:` ***Tampon***

Le Tampon objet qui contient les octets de charge utile bruts du certificat client.

`policyConstraints:` ***Objet***

Objet contenant des informations provenant de l' extension Policy Constraints, qui spécifie les contraintes de validation pour les certificats CA.

`requireExplicitPolicy:` ***Numéro***

Spécifie le nombre maximum de certificats adjacents dans la chaîne qui n'ont pas besoin de spécifier de politique explicite.

`inhibitPolicyMapping:` ***Numéro***

Spécifie le nombre maximum de certificats adjacents dans la chaîne de certificats avant que les mappages de politiques ne soient ignorés.

`publicKeyCurveName:` ***Corde***

Nom de la courbe elliptique standard sur laquelle est basée la cryptographie de la clé publique. Cette valeur est déterminée par l'OID ou les paramètres de courbe explicites spécifiés dans le certificat.

`publicKeyExponent:` ***Corde | Null***

Une représentation hexadécimale sous forme de chaîne de l'exposant de clé publique.

`publicKeyHasExplicitCurve:` ***Booléen | Null***

Indique si le certificat spécifie des paramètres explicites pour la courbe elliptique de la clé publique.

`publicKeyModulus:` ***Corde | Null***

Une représentation hexadécimale sous forme de chaîne du module de clé publique, telle que `010001`.

`policyMappings:` ***Tableau d' objets***

Tableau d'objets contenant des informations provenant de l'extension Policy Mappings, qui indique des politiques équivalentes les unes aux autres. Chaque objet contient les champs suivants.

`issuerDomainPolicy:` ***Corde***

L'OID de la politique de l'émetteur.

`subjectDomainPolicy:` ***Corde***

L'OID de la politique en question.

`signatureAlgorithm:` ***Corde | Null***

Algorithme appliqué pour signer le certificat client. Le tableau suivant présente certaines des valeurs possibles :

RFC	Algorithme
RFC 3279	• md2WithRSAEncryption • md5WithRSAEncryption • sha1WithRSAEncryption
RFC 4055	• sha224WithRSAEncryption • sha256WithRSAEncryption • sha384WithRSAEncryption • sha512WithRSAEncryption
RFC 4491	• id-GostR3411-94-with-Gost3410-94 • id-GostR3411-94-with-Gost3410-2001

`subject:` ***Corde***

Le nom commun du sujet (CN) du certificat client.

`subjectAlternativeNames:` ***Array***

Tableau de chaînes correspondant aux noms alternatifs de sujet (SAN) inclus dans le certificat client. Les SAN pris en charge sont les noms DNS, les adresses e-mail, les URI et les adresses IP.

`subjectDistinguishedName:` ***Objet***

Objet contenant des informations sur le nom distinctif du sujet du certificat. Chaque objet contient les propriétés suivantes :

`commonName:` ***Corde***

Le nom commun (CN).

`country:` ***Tableau de cordes***

Le nom du pays (C).

`emailAddress:` ***Corde***

L'adresse e-mail.

`organization:` ***Tableau de cordes***

Le nom de l'organisation (O).

`organizationalUnit:` ***Tableau de cordes***

Le nom de l'unité organisationnelle (OU).

`locality:` ***Tableau de cordes***

Le nom de la localité (L).

`stateOrProvince:` ***Tableau de cordes***

Le nom de l'État ou de la province (ST).

`subjectKeyIdentifier:` ***Corde***

Identifiant de la clé publique du sujet du certificat client, exprimé sous forme de chaîne d'octets.

`clientCertificates:` ***Tableau d'objets***

Tableau d'objets de certificat pour chaque certificat client TLS intermédiaire. Le certificat d' entité finale, également appelé certificat feuille, est le premier objet du tableau ; cet objet est également renvoyé par le `clientCertificate` propriété.

`clientCertificateRequested:` ***Booléen***

La valeur est `true` si le serveur TLS a demandé un certificat client.

Accès uniquement sur `SSL_OPEN`, `SSL_ALERT`, ou `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`clientExtensions:` ***Array | Nul***

Tableau d'objets d'extension client contenant les propriétés suivantes :

`id:` ***Numéro***

Numéro d'identification de l'extension client TLS.

`length:` ***Numéro***

Longueur totale de l'extension client TLS, exprimée en octets.

Note: Une extension peut être tronquée si sa longueur dépasse la taille maximale. La valeur par défaut est de 512 octets. Une troncature s'est produite si la valeur de cette propriété est inférieure à la mémoire tampon renvoyée par le `getClientExtensionData()` méthode.

`name:` ***Corde***

Le nom de l'extension du client TLS, s'il est connu. Sinon, la valeur indique que l'extension est inconnue. Consultez le tableau des extensions TLS connues dans le Section des méthodes.

Accès uniquement sur `SSL_OPEN` ou `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`clientExtensionsHex:` ***Corde***

Représentation hexadécimale de la liste triée des extensions clients.

Note: Les valeurs GREASE (Generate Random Extensions And Sustain Extensibility) sont supprimées de la liste avant l'encodage.

Accès uniquement sur `SSL_OPEN` et `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`clientHelloVersion:` ***Numéro***

Version de TLS spécifiée par le client dans le paquet Hello du client.

`clientL2Bytes:` ***Numéro***

Le nombre de L2 octets envoyés par le client depuis la dernière `SSL_RECORD` événement.

Accès uniquement sur `SSL_RECORD` ou `SSL_CLOSE` événements ; sinon, une erreur se produira.

`clientPkts:` ***Numéro***

Le nombre de paquets envoyés par le client depuis le dernier `SSL_RECORD` événement.

Accès uniquement sur `SSL_RECORD` ou `SSL_CLOSE` événements ; sinon, une erreur se produira.

`clientSessionId:` ***Corde***

L'ID de session du client sous forme de tableau d'octets codé sous forme de chaîne.

`clientZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par le client depuis la dernière `SSL_RECORD` événement.

Accès uniquement sur `SSL_RECORD` ou `SSL_CLOSE` événements ; sinon, une erreur se produira.

`contentType:` ***Corde***

Type de contenu pour l'enregistrement en cours.

Accès uniquement sur `SSL_RECORD` événements ; sinon, une erreur se produira.

`ecPointFormatsHex:` ***Corde***

Représentation hexadécimale des formats de points de courbe elliptique que le client peut analyser.

Accès uniquement sur `SSL_OPEN` et `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`encryptionProtocol:` ***Corde***

Version du protocole TLS avec laquelle la transaction est chiffrée.

`handshakeTime:` ***Numéro***

Temps nécessaire pour négocier la connexion TLS, exprimé en millisecondes. Plus précisément, le délai entre le moment où le client envoie un `ClientBonjour` message et le serveur envoie `ChangeCipherSpec` valeurs telles que spécifiées dans la RFC 2246.

Accès uniquement sur `SSL_OPEN` ou `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`heartbeatPayloadLength:` ***Numéro***

La valeur du champ de longueur de charge utile de la structure de données HeartbeatMessage telle que spécifiée dans la RFC 6520.

Accès uniquement sur `SSL_HEARTBEAT` événements ; sinon, une erreur se produira.

`heartbeatType:` ***Numéro***

Représentation numérique du champ HeartbeatMessageType de la structure de données HeartbeartMessage telle que spécifiée dans la RFC 6520. Les valeurs valides sont `SSL.HEARTBEAT_TYPE_REQUEST (1)`, `SSL.HEARTBEAT_TYPE_RESPONSE (2)`, ou `SSL.HEARTBEAT_TYPE_UNKNOWN (255)`.

Accès uniquement sur `SSL_HEARTBEAT` événements ; sinon, une erreur se produira.

`host:` ***Corde | Nul***

L'indication du nom du serveur TLS (SNI), si disponible.

Accès uniquement sur `SSL_OPEN` ou `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`isAborted:` ***Booléen***

La valeur est `true` si la session TLS est abandonnée.

Accès uniquement sur `SSL_CLOSE`, `SSL_OPEN`, et `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`isCompressed:` ***Booléen***

La valeur est `true` si l'enregistrement TLS est compressé.

`isDecrypted:` ***Booléen***

La valeur est `true` si le système ExtraHop a déchiffré et analysé la transaction de manière sécurisée. L'analyse du trafic déchiffré peut révéler des menaces avancées qui se cachent dans le trafic chiffré.

`isEncrypted:` ***Booléen***

La valeur est `true` si la connexion TLS est cryptée.

`isResumed:` ***Booléen***

La valeur est `true` si la connexion est reprise à partir d'une session TLS existante et qu'il ne s'agit pas d'une nouvelle session TLS.

Accès uniquement sur `SSL_OPEN`, `SSL_CLOSE`, `SSL_ALERT`, `SSL_HEARTBEAT`, ou `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`isStartTLS:` ***Booléen***

La valeur est `true` si la négociation de la session TLS a été initiée par le mécanisme STARTTLS du protocole.

Accès uniquement sur SSL_OPEN, SSL_CLOSE, SSL_ALERT, SSL_HEARTBEAT, ou SSL_RENEGOTIATE événements ; sinon, une erreur se produira.

`isV2ClientHello:` ***Booléen***

La valeur est `true` si l'enregistrement Hello correspond à SSLv2.

`isWeakCipherSuite:` ***Booléen***

La valeur est `true` si la suite de chiffrement chiffrant la session TLS est considérée comme faible. Les suites de chiffrement NULL, anonymous et EXPORT sont considérées comme faibles, tout comme les suites chiffrant avec CBC, DES, 3DES, MD5 ou RC4.

Accès uniquement sur SSL_OPEN, SSL_CLOSE, SSL_ALERT, SSL_HEARTBEAT, ou SSL_RENEGOTIATE événements ; sinon, une erreur se produira.

`ja3Text:` ***Corde | Nul***

La chaîne JA3 complète pour le client, y compris la version TLS du client hello, les chiffrements acceptés, les extensions SSL, les courbes elliptiques et les formats de courbes elliptiques.

`ja3Hash:` ***Corde | Nul***

Le hachage MD5 de la chaîne JA3 pour le client.

`ja3sText:` ***Corde | Nul***

La chaîne JA3S complète du serveur, y compris la version SSL du serveur Hello, les chiffrements acceptés et les extensions TLS.

`ja3sHash:` ***Corde | Nul***

Le hachage MD5 de la chaîne JA3S pour le serveur.

`ja4Fingerprint:` ***Corde | Nul***

L'empreinte digitale JA4 complète du client, qui comprend les informations suivantes :

- Le protocole de la couche de transport (L4)
- La version TLS
- Si l'extension SNI (Server Name Indicator) a été spécifiée
- Le nombre de suites de chiffrement
- Le nombre de prolongations
- La première valeur ALPN (Application Layer Protocol Negotiation) répertoriée
- Le hachage SHA256 tronqué des suites de chiffrement
- Le hachage SHA256 tronqué des extensions

`privateKeyId:` ***Corde | Nul***

ID de chaîne associé à la clé privée si le système ExtraHop déchiffre le trafic TLS. La valeur est `null` si le système ExtraHop ne déchiffre pas le trafic SSL.

Pour trouver l'identifiant de la clé privée dans les paramètres d'administration, cliquez sur **Capturez** à partir du Configuration du système section, cliquez sur **Décryptage SSL**, puis cliquez sur un certificat. La fenêtre contextuelle affiche tous les identifiants du certificat.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `SSL.commitRecord()` sur l'un ou l'autre SSL_OPEN, SSL_CLOSE, SSL_ALERT, SSL_HEARTBEAT, ou SSL_RENEGOTIATE événement.

L'événement pour lequel la méthode a été appelée détermine les propriétés que l'objet d'enregistrement par défaut peut contenir, comme indiqué dans le tableau suivant :

Événement	Propriétés disponibles
SSL_ALERT	• `alertCode` • `alertLevel` • `certificateFingerprint` • `certificateIsSelfSigned`

Événement	Propriétés disponibles
	• certificateIssuer • certificateKeySize • certificateNotAfter • certificateNotBefore • certificateSignatureAlgorithm • certificateSubject • cipherSuite • clientAddr • clientBytes • clientCertificateRequested • clientIsExternal • clientL2Bytes • clientPkts • clientPort • clientRTO • clientZeroWnd • isCompressed • isWeakCipherSuite • proto • receiverIsExternal • reqBytes • reqL2Bytes • reqPkts • reqRTO • rspBytes • rspL2Bytes • rspPkts • rspRTO • senderIsExternal • serverAddr • serverBytes • serverIsExternal • serverL2Bytes • serverPkts • serverPort • serverRTO • serverZeroWnd • version
SSL_CLOSE	• certificateIsSelfSigned • certificateIssuer • certificateFingerprint • certificateKeySize • certificateNotAfter • certificateNotBefore • certificateSignatureAlgorithm • certificateSubject • cipherSuite • clientAddr

Événement	Propriétés disponibles
	• clientBytes • clientIsExternal • clientL2Bytes • clientPkts • clientPort • clientRTO • clientZeroWnd • isAborted • isCompressed • isWeakCipherSuite • proto • receiverIsExternal • reqBytes • reqPkts • reqL2Bytes • reqRTO • rspBytes • rspL2Bytes • rspPkts • rspRTO • senderIsExternal • serverAddr • serverBytes • serverIsExternal • serverL2Bytes • serverPkts • serverPort • serverRTO • serverZeroWnd • version
SSL_HEARTBEAT	• certificateFingerprint • certificateIssuer • certificateKeySize • certificateNotAfter • certificateNotBefore • certificateSignatureAlgorithm • certificateSubject • cipherSuite • clientIsExternal • clientZeroWnd • heartbeatPayloadLength • heartbeatType • isCompressed • receiverIsExternal • senderIsExternal • serverIsExternal • serverZeroWnd

Événement	Propriétés disponibles
	• version
SSL_OPEN	• certificateFingerprint • certificateIsSelfSigned • certificateIssuer • certificateKeySize • certificateNotAfter • certificateNotBefore • certificateSignatureAlgorithm • certificateSubject • certificateSubjectAlternativeNames • cipherSuite • clientAddr • clientAlpn • clientBytes • clientCertificateRequested • clientIsExternal • clientL2Bytes • clientPkts • clientPort • clientRTO • clientZeroWnd • handshakeTime • host • isAborted • isCompressed • isRenegotiate • isWeakCipherSuite • ja3Hash • ja3sHash • ja4Fingerprint • proto • receiverIsExternal • reqBytes • reqL2Bytes • reqPkts • reqRTO • rspBytes • rspL2Bytes • rspPkts • rspRTO • senderIsExternal • serverAddr • serverAlpn • serverBytes • serverIsExternal • serverL2Bytes • serverPkts • serverPort

Événement	Propriétés disponibles
	• serverRTO • serverZeroWnd • version
SSL_RENEGOTIATE **Note:** Le SSL_OPEN le format d'enregistrement est appliqué aux enregistrements validés lors de cet événement.	• certificateFingerprint • certificateKeySize • certificateNotAfter • certificateNotBefore • certificateSignatureAlgorithm • certificateSubject • cipherSuite • clientAlpn • clientIsExternal • handshakeTime • host • isAborted • isCompressed • receiverIsExternal • senderIsExternal • serverAlpn • serverIsExternal • version

`recordLength:` ***Numéro***

La valeur du champ de longueur du `TLSPlaintext`, `TLSCompressed`, et `TLSCiphertext` structures de données telles que spécifiées dans la RFC 5246.

Accès uniquement sur `SSL_RECORD`, `SSL_ALERT`, ou `SSL_HEARTBEAT` événements ; sinon, une erreur se produira.

`recordType:` ***Numéro***

La représentation numérique du champ de type du `TLSPlaintext`, `TLSCompressed`, et `TLSCiphertext` structures de données telles que spécifiées dans la RFC 5246.

Accès uniquement sur `SSL_RECORD`, `SSL_ALERT`, et `SSL_HEARTBEAT` événements ; sinon, une erreur se produira.

`roundTripTime:` ***Numéro***

Temps médian aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

Accès uniquement sur `SSL_RECORD` ou `SSL_CLOSE` événements ; sinon, une erreur se produira.

`serverExtensions:` ***Array | Nul***

Tableau d'objets d'extension de serveur contenant les propriétés suivantes :

`id:` ***Numéro***

Numéro d'identification de l'extension de serveur SSL.

`length:` ***Numéro***

Longueur totale de l'extension de serveur SSL, exprimée en octets.

Note: Une extension peut être tronquée si sa longueur dépasse la taille maximale. La valeur par défaut est de 512 octets. Une troncature s'est produite si la valeur de cette propriété est inférieure à la mémoire tampon renvoyée par le `getClientExtensionData()` méthode.

`name:` ***Corde***

Le nom de l'extension du serveur TLS, s'il est connu. Sinon, la valeur indique que l'extension est inconnue. Consultez le tableau des extensions TLS connues dans le Section des méthodes.

Accès uniquement sur `SSL_OPEN` ou `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`serverExtensionsHex:` ***Corde***

Représentation hexadécimale de la liste triée des extensions de serveur.

Note: Les valeurs GREASE (Generate Random Extensions And Sustain Extensibility) sont supprimées de la liste avant l'encodage.

Accès uniquement sur `SSL_OPEN` et `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`serverBytes:` ***Numéro***

Le nombre d'octets envoyés par le serveur depuis la dernière `SSL_RECORD` événement.

Accès uniquement sur `SSL_RECORD` ou `SSL_CLOSE` événements ; sinon, une erreur se produira.

`serverHelloVersion:` ***Numéro***

Version de TLS spécifiée par le serveur dans le paquet Hello du serveur.

`serverL2Bytes:` ***Numéro***

Le nombre de L2 octets envoyés par le serveur depuis la dernière `SSL_RECORD` événement.

Accès uniquement sur `SSL_RECORD` ou `SSL_CLOSE` événements ; sinon, une erreur se produira.

`serverPkts:` ***Numéro***

Le nombre de paquets envoyés par le serveur depuis la dernière `SSL_RECORD` événement.

Accès uniquement sur `SSL_RECORD` ou `SSL_CLOSE` événements ; sinon, une erreur se produira.

`serverSessionId:` ***Corde***

Tableau d'octets d'ID de session du serveur, codé sous forme de chaîne.

`serverZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par le serveur depuis la dernière `SSL_RECORD` événement.

Accès uniquement sur `SSL_RECORD` ou `SSL_CLOSE` événements ; sinon, une erreur se produira.

`startTLSProtocol:` ***Corde | Nul***

Le protocole à partir duquel le client a envoyé une commande STARTTLS.

`supportedGroupsHex:` ***Corde***

Représentation hexadécimale des groupes Diffie-Hellman (ECDH) à courbe elliptique pris en charge par le client.

Accès uniquement sur `SSL_OPEN` et `SSL_RENEGOTIATE` événements ; sinon, une erreur se produira.

`version:` ***Numéro***

Version du protocole SSL avec le numéro de version hexadécimal RFC, exprimé sous forme décimale.

Version	Hex	Décimal
SSLv2	0x200	2
SSLv3	0x300	768
TLS 1.0	0x301	769
TLS 1.1	0x302	770
TLS 1.2	0x303	771

Version	Hex	Décimal
TLS 1.3	0x304	772

TCP

Le TCP la classe vous permet d'accéder aux propriétés et de récupérer des métriques à partir d'événements TCP et de FLOW_TICK et FLOW_TURN événements.

Le FLOW_TICK et FLOW_TURN les événements sont définis dans le Flow section.

Évènements

TCP_CLOSE

S'exécute lorsque la connexion TCP est interrompue en raison de sa fermeture, de son expiration ou de son abandon.

TCP_OPEN

S'exécute lorsque la connexion TCP est complètement établie pour la première fois.

Le FLOW_CLASSIFY l'événement se déroule après le TCP_OPEN événement visant à déterminer L7 protocole du flux TCP.

Note: Si une connexion TCP est bloquée pendant une longue période, l'événement TCP_OPEN s'exécute à nouveau lorsque la connexion reprend. Les propriétés et méthodes TCP suivantes sont nulles lorsque l' événement est exécuté pour rétablir la connexion :

- getOption
- handshakeTime
- hasECNEcho
- hasECNEcho1
- hasECNEcho2
- initRcvWndSize
- initRcvWndSize1
- initRcvWndSize2
- initSeqNum
- initSeqNum1
- initSeqNum2
- options
- options1
- options2

TCP_PAYLOAD

S'exécute lorsque la charge utile correspond aux critères configurés dans le déclencheur associé .

En fonction du Flow , la charge utile TCP se trouve dans les propriétés suivantes :

- Flow.client.payload
- Flow.payload1
- Flow.payload2
- Flow.receiver.payload
- Flow.sender.payload
- Flow.server.payload

Des options de charge utile supplémentaires sont disponibles lorsque vous créez un déclencheur qui s'exécute lors de cet événement. Voir Options de déclencheur avancées pour plus d' informations.

Méthodes

`getOption(kind:` ***Numéro*** `):` ***Objet*** | ***Null***

Renvoie un objet d'option TCP qui correspond au type d'option spécifié. Pour obtenir la liste des types d'options valides, voir Options TCP. Spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.getOption(1)` ou `TCP.server.getOption(1)`.

S'applique uniquement à `TCP_OPEN` événements.

Propriétés

`handshakeTime:` ***Numéro***

Durée nécessaire pour négocier la connexion TCP, exprimée en millisecondes.

Accès uniquement sur `TCP_OPEN` événements ; dans le cas contraire, une erreur se produira.

`hasECNEcho:` ***Booléen***

La valeur est `true` si le drapeau ECN est activé sur un équipement pendant l' établissement dacp tridirectionnel. Spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.hasECNEcho` ou `TCP.server.hasECNEcho`.

Accès uniquement sur `TCP_OPEN` événements ; dans le cas contraire, une erreur se produira.

`hasECNEcho1:` ***Booléen***

La valeur est `true` si le drapeau ECN est activé pendant l'établissement dacp tridirectionnel associé à l'un des deux appareils de la connexion ; l'autre équipement est représenté par `hasECNEcho2`. L'équipement représenté par `hasECNEcho1` reste cohérent pour la connexion.

Accès uniquement sur `TCP_OPEN` événements ; dans le cas contraire, une erreur se produira.

`hasECNEcho2:` ***Booléen***

La valeur est `true` si le drapeau ECN est activé pendant l'établissement dacp tridirectionnel associé à l'un des deux appareils de la connexion ; l'autre équipement est représenté par `hasECNEcho1`. L'équipement représenté par `hasECNEcho2` reste cohérent pour la connexion.

Accès uniquement sur `TCP_OPEN` événements ; dans le cas contraire, une erreur se produira.

`initRcvWndSize:` ***Numéro***

Taille initiale de la fenêtre coulissante TCP sur un équipement négociée lors de l'établissement dacp tridimensionnel . Spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.initRcvWndSize` ou `TCP.server.initRcvWndSize`.

Accès uniquement sur `TCP_OPEN` événements ; dans le cas contraire, une erreur se produira.

`initRcvWndSize1:` ***Numéro***

La taille initiale de la fenêtre coulissante TCP négociée lors de l'établissement dacp tridirectionnel associé à l'un des deux périphériques de la connexion ; l'autre équipement est représenté par `initRcvWndSize2`. L'équipement représenté par `initRcvWndSize1` reste cohérent pour la connexion.

Accès uniquement sur `TCP_OPEN` événements ; dans le cas contraire, une erreur se produira.

`initRcvWndSize2:` ***Numéro***

La taille initiale de la fenêtre coulissante TCP négociée lors de l'établissement dacp tridirectionnel associé à l'un des deux périphériques de la connexion ; l'autre équipement est représenté par `initRcvWndSize1`. L'équipement représenté par `initRcvWndSize2` reste cohérent pour la connexion.

Accès uniquement sur `TCP_OPEN` événements ; dans le cas contraire, une erreur se produira.

`initSeqNum:` ***Numéro***

Numéro de séquence initial envoyé par un équipement lors de l'établissement dacp tridimensionnel. Spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.initSeqNum` ou `TCP.server.initSeqNum`.

Accès uniquement sur TCP_OPEN événements ; dans le cas contraire, une erreur se produira.

`initSeqNum1:` ***Numéro***

Numéro de séquence initial lors de l'établissement dune liaison à trois associés à l'un des deux appareils de la connexion ; l'autre équipement est représenté par `initSeqNum2`. L'équipement représenté par `initSeqNum1` reste cohérent pour la connexion.

Accès uniquement sur TCP_OPEN événements ; dans le cas contraire, une erreur se produira.

`initSeqNum2:` ***Numéro***

Numéro de séquence initial lors de l'établissement dune liaison à trois associés à l'un des deux appareils de la connexion ; l'autre équipement est représenté par `initSeqNum1`. L'équipement représenté par `initSeqNum2` reste cohérent pour la connexion.

Accès uniquement sur TCP_OPEN événements ; dans le cas contraire, une erreur se produira.

`isAborted:` ***Booléen***

La valeur est `true` si un flux TCP a été abandonné par le biais d'une réinitialisation TCP (RST) avant l'arrêt de la connexion. Le flux peut être interrompu par un équipement. Le cas échéant, spécifiez le rôle de l'équipement dans la syntaxe, par exemple, `TCP.client.isAborted` ou `TCP.server.isAborted`.

Cette condition peut être détectée dans n'importe quel événement TCP et dans tout événement L7 impacté (par exemple, HTTP_REQUEST ou DB_RESPONSE).

Note:
- Un L4 l'abandon se produit lorsqu'une connexion TCP est fermée par un RST au lieu d'un arrêt progressif.
- Un L7 l'abandon de réponse se produit lorsqu'une connexion se ferme au milieu d'une réponse. Cela peut être dû à un RST, à un arrêt progressif du FIN ou à une expiration.
- Un abandon de demande L7 se produit lorsqu'une connexion se ferme au milieu d'une demande. Cela peut également être dû à un RST, à un arrêt progressif du FIN ou à une expiration.

`isExpired:` ***Booléen***

La valeur est `true` si la connexion TCP a expiré au moment de l' événement. Le cas échéant, spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.isExpired` ou `TCP.server.isExpired`.

Accès uniquement sur TCP_CLOSE événements ; dans le cas contraire, une erreur se produira.

`isReset:` ***Booléen***

La valeur est `true` si une réinitialisation TCP (RST) a été détectée alors que la connexion était en train d'être interrompue.

`nagleDelay:` ***Numéro***

Le nombre de retards Nagle associés à un équipement dans le flux. Spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.nagleDelay` ou `TCP.server.nagleDelay`.

Accès uniquement sur FLOW_TICK et FLOW_TURN événements ; dans le cas contraire, une erreur se produira.

`nagleDelay1:` ***Numéro***

Le nombre de retards Nagle associés à l'un des deux appareils du flux ; l'autre appareil est représenté par `nagleDelay1`. L'équipement représenté par `nagleDelay2` reste cohérent pour la connexion.

Accès uniquement sur FLOW_TICK et FLOW_TURN événements ; dans le cas contraire, une erreur se produira.

`nagleDelay1:` ***Numéro***

Le nombre de retards Nagle associés à l'un des deux appareils du flux ; l'autre appareil est représenté par `nagleDelay2`. L'équipement représenté par `nagleDelay1` reste cohérent pour la connexion.

Accès uniquement sur FLOW_TICK et FLOW_TURN événements ; dans le cas contraire, une erreur se produira.

`options:` ***Array***

Tableau d'objets représentant les options TCP d'un équipement dans les paquets d'établissement dacp nitial d'établissement d'une liaison. Spécifiez le TCP client ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.options` ou `TCP.server.options`. Pour plus d'informations, consultez la section sur les options TCP ci-dessous.

Accès uniquement sur TCP_OPEN événements ; dans le cas contraire, une erreur se produira.

`options1:` ***Array***

Un tableau d'options représentant les options TCP dans les paquets d'établissement d'une liaison initiaux associés à l'un des deux périphériques de la connexion ; l'autre équipement est représenté par `options2`. L'équipement représenté par `options1` reste cohérent pour la connexion. Pour plus d'informations, consultez la section sur les options TCP ci-dessous.

Accès uniquement sur TCP_OPEN événements ; dans le cas contraire, une erreur se produira.

`options2:` ***Array***

Un tableau d'options représentant les options TCP dans les paquets d'établissement d'une liaison initiaux associés à l'un des deux périphériques de la connexion ; l'autre équipement est représenté par `options1`. L'équipement représenté par `options2` reste cohérent pour la connexion. Pour plus d'informations, consultez la section sur les options TCP ci-dessous.

Accès uniquement sur TCP_OPEN événements ; dans le cas contraire, une erreur se produira.

`overlapSegments:` ***Numéro***

Nombre de segments TCP non identiques, transmis par un équipement dans le flux, où deux segments TCP ou plus contiennent des données pour la même partie du flux. Spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.overlapSegments` ou `TCP.server.overlapSegments`.

Accès uniquement sur FLOW_TICK ou FLOW_TURN événements ; dans le cas contraire, une erreur se produira.

`overlapSegments1:` ***Numéro***

Le nombre de segments TCP non identiques où deux segments ou plus contiennent des données pour la même partie du flux. Les segments TCP sont transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `overlapSegments2`. L'équipement représenté par `overlapSegments1` reste constant pour le flux.

Accès uniquement sur FLOW_TICK ou FLOW_TURN événements ; dans le cas contraire, une erreur se produira.

`overlapSegments2:` ***Numéro***

Le nombre de segments TCP non identiques où deux segments ou plus contiennent des données pour la même partie du flux. Les segments TCP sont transmis par l'un des deux périphériques du flux ; l'autre équipement est représenté par `overlapSegments1`. L'équipement représenté par `overlapSegments2` reste constant pour le flux.

Accès uniquement sur FLOW_TICK ou FLOW_TURN événements ; dans le cas contraire, une erreur se produira.

`rcvWndThrottle:` ***Numéro***

Le nombre de limiteurs de fenêtre de réception envoyés par un équipement dans le flux. Spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.rcvWndThrottle` ou `TCP.server.rcvWndThrottle`.

Accès uniquement sur FLOW_TICK et FLOW_TURN événements ; dans le cas contraire, une erreur se produira.

`rcvWndThrottle1:` ***Numéro***

Le nombre de limiteurs de fenêtre de réception envoyés par l'un des deux appareils du flux ; l'autre équipement est représenté par `rcvWndThrottle2`. L'équipement représenté par `rcvWndThrottle1` reste cohérent pour la connexion.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; dans le cas contraire, une erreur se produira.

`rcvWndThrottle2:` ***Numéro***

Le nombre de limiteurs de fenêtre de réception envoyés par l'un des deux appareils du flux ; l'autre équipement est représenté par `rcvWndThrottle1`. L'équipement représenté par `rcvWndThrottle2` reste cohérent pour la connexion.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; dans le cas contraire, une erreur se produira.

`retransBytes:` ***Numéro***

Nombre d'octets retransmis via TCP par un client ou un équipement serveur dans le flux. Spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.retransBytes` ou `TCP.server.retransBytes`.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; dans le cas contraire, une erreur se produira.

`retransBytes1:` ***Numéro***

Le nombre d'octets retransmis via TCP par l'un des deux périphériques du flux ; l'autre équipement est représenté par `retransBytes2`. L'équipement représenté par `retransBytes1` reste cohérent pour la connexion.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; dans le cas contraire, une erreur se produira.

`retransBytes2:` ***Numéro***

Le nombre d'octets retransmis via TCP par l'un des deux périphériques du flux ; l'autre équipement est représenté par `retransBytes1`. L'équipement représenté par `retransBytes2` reste cohérent pour la connexion.

Accès uniquement sur `FLOW_TICK` ou `FLOW_TURN` événements ; dans le cas contraire, une erreur se produira.

`zeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par un équipement dans le flux. Spécifiez le client TCP ou le serveur TCP dans la syntaxe, par exemple, `TCP.client.zeroWnd` ou `TCP.server.zeroWnd`.

Accès uniquement sur `FLOW_TICK` et `FLOW_TURN` événements ; dans le cas contraire, une erreur se produira.

`zeroWnd1:` ***Numéro***

Le nombre de fenêtres nulles envoyées par l'un des deux appareils du flux ; l'autre équipement est représenté par `zeroWnd2`. L'équipement représenté par `zeroWnd1` reste cohérent pour la connexion.

Accès uniquement sur `FLOW_TICK` et `FLOW_TURN` événements ; dans le cas contraire, une erreur se produira.

`zeroWnd2:` ***Numéro***

Le nombre de fenêtres nulles envoyées par l'un des deux appareils du flux ; l'autre équipement est représenté par `zeroWnd1`. L'équipement représenté par `zeroWnd2` reste cohérent pour la connexion.

Accès uniquement sur `FLOW_TICK` et `FLOW_TURN` événements ; dans le cas contraire, une erreur se produira.

Options TCP

Tous les objets TCP Options possèdent les propriétés suivantes :

`kind`: ***Numéro***

Le numéro de type de l'option TCP.

Numéro de type	Signification
0	End of Option List
1	No-Operation
2	Maximum Segment Size
3	Window Scale
4	SACK Permitted
5	SACK
6	Echo (obsoleted by option 8)
7	Echo Reply (obsoleted by option 8)
8	Timestamps
9	Partial Order Connection Permitted (obsolete)
10	Partial Order Service Profile (obsolete)
11	CC (obsolete)
12	CC.NEW (obsolete)
13	CC.ECHO (obsolete)
14	TCP Alternate Checksum Request (obsolete)
15	TCP Alternate Checksum Data (obsolete)
16	Skeeter
17	Bubba
18	Trailer Checksum Option
19	MD5 Signature Option (obsoleted by option 29)
20	SCPS Capabilities
21	Selective Negative acknowledgments
22	Record Boundaries
23	Corruption experienced
24	SNAP
25	Unassigned (released 2000-12-18)
26	TCP Compression Filter
27	Quick-Start Response
28	User Timeout Option (also, other known authorized use)
29	TCP Authentication Option (TCP-AO)

Numéro de type	Signification
30	Multipath TCP (MPTCP)
31	Reserved (known authorized used without proper IANA assignment)
32	Reserved (known authorized used without proper IANA assignment)
33	Reserved (known authorized used without proper IANA assignment)
34	TCP Fast Open Cookie
35-75	Reserved
76	Reserved (known authorized used without proper IANA assignment)
77	Reserved (known authorized used without proper IANA assignment)
78	Reserved (known authorized used without proper IANA assignment)
79-252	Reserved
253	RFC3692-style Experiment 1 (also improperly used for shipping products)
254	RFC3692-style Experiment 2 (also improperly used for shipping products)

`name:` ***Corde***

Nom de l'option TCP.

La liste suivante contient les noms des options TCP courantes et leurs propriétés spécifiques :

Taille maximale du segment (nom « mss », type d'option 2)

`value:` ***Numéro***

Taille maximale du segment.

Window Scale (nom « wscale », type 3)

`value:` ***Numéro***

Le facteur d'échelle de la fenêtre.

Accusé de réception sélectif autorisé (nom « sack-permitted », type 4)

Aucune propriété supplémentaire. Sa présence indique que l'option d' accusé de réception sélectif a été incluse dans le SYN.

Horodatage (nom « horodateur », type 8)

`tsval:` ***Numéro***

Le champ TSVal pour l'option.

`tsecr:` ***Numéro***

Le champ TSecr pour l'option.

Réponse de démarrage rapide (nom « quickstart-rsp », type 27)

`rate-request:` ***Numéro***

Débit demandé pour le transport, exprimé en octets par seconde.

`ttl-diff:` ***Numéro***

Le TTLDif.

qs-`nonce:` ***Numéro***

Le QS Nonce.

Adresse Akamai (nom « akamai-addr », type 28)

`value:` ***Adresse iPad***

Adresse IP du serveur Akamai.

User Timeout (nom « user-timeout », type 28)

`value:` ***Numéro***

Le délai d'expiration de l'utilisateur.

Authentification (nom « tcp-ao », type 29)

`keyId property:` ***Numéro***

L'identifiant de la clé utilisée.

`rNextKeyId:` ***Numéro***

L'identifiant de clé pour l'identifiant de clé « receive next ».

`mac:` ***Tampon***

Le code d'authentification du message.

Multipath (nom « mptcp », type 30)

`value:` ***Tampon***

La valeur multipath.

Note: Les options d'adresse d'Akamai et de délai d'expiration de l'utilisateur sont différenciées par la durée de l'option.

Voici un exemple d'options TCP :

```
if (TCP.client.options != null) {

   var optMSS = TCP.client.getOption(2)

   if (optMSS && (optMSS.value > 1460)) {
       Network.metricAddCount('large_mss', 1);
       Network.metricAddDetailCount('large_mss_by_client_ip',
                                    Flow.client.ipaddr + " " + optMSS.value,
 1);
   }

}
```

`Telnet`

Le `Telnet` la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `TELNET_MESSAGE` événements.

Évènements

`TELNET_MESSAGE`

S'exécute sur une commande telnet ou une ligne de données provenant du telnet client ou serveur.

Méthodes

`commitRecord():` ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `TELNET_MESSAGE` événement.

Pour consulter les propriétés par défaut attribuées à l'objet d'enregistrement, consultez le `record` propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si `commitRecord()` méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

`command:` ***Corde***

Type de commande. La valeur est `null` si l'événement a été lancé en raison de l'envoi d'une ligne de données.

Les valeurs suivantes sont valides :

- `Abort`
- `Abort Output`
- `Are You There`
- `Break`
- `Data Mark`
- `DO`
- `DON'T`
- `End of File`
- `End of Record`
- `Erase Character`
- `Erase Line`
- `Go Ahead`
- `Interrupt Process`
- `NOP`
- `SB`
- `SE`
- `Suspend`
- `WILL`
- `WON'T`

`line:` ***Corde***

Une ligne des données envoyées par le client ou serveur. Les séquences d'échappement des terminaux et les caractères spéciaux sont filtrés. Le mouvement du curseur et l' édition de lignes ne sont pas simulés, sauf pour les caractères de retour arrière.

`option:` ***Corde***

L'option en cours de négociation. La valeur est `null` si l'option n'est pas valide. Les valeurs suivantes sont valides :

- `3270-REGIME`
- `AARD`
- `ATCP`
- `AUTHENTICATION`
- `BM`
- `CHARSET`
- `COM-PORT-OPTION`

- DET
- ECHO
- ENCRYPT
- END-OF-RECORD
- ENVIRON
- EXPOPL
- EXTEND-ASCII
- FORWARD-X
- GMCP
- KERMIT
- LINEMODE
- LOGOUT
- NAOCRD
- NAOFFD
- NAOHTD
- NAOHTS
- NAOL
- NAOLFD
- NAOP
- NAOVTD
- NAOVTS
- NAWS
- NEW-ENVIRON
- OUTMRK
- PRAGMA-HEARTBEAT
- PRAGMA-LOGON
- RCTE
- RECONNECT
- REMOTE-SERIAL-PORT
- SEND-LOCATION
- SEND-URL
- SSPI-LOGON
- STATUS
- SUPDUP
- SUPDUP-OUTPUT
- SUPPRESS-GO-AHEAD
- TERMINAL-SPEED
- TERMINAL-TYPE
- TIMING-MARK
- TN3270E
- TOGGLE-FLOW-CONTROL
- TRANSMIT-BINARY
- TTYLOC
- TUID
- X-DISPLAY-LOCATION
- X.3-PAD
- XAUTH

`optionData`: ***Tampon***

Pour les sous-négociations d'options (commande SB), les données brutes spécifiques à l'option sont envoyées. La valeur est `null` si la commande n'est pas SB.

`record:` ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `Telnet.commitRecord()` sur un `TELNET_MESSAGE` événement.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientIsExternal`
- `command`
- `option`
- `receiverBytes`
- `receiverIsExternal`
- `receiverL2Bytes`
- `recieverPkts`
- `receiverRTO`
- `receiverZeroWnd`
- `roundTripTime`
- `senderBytes`
- `senderIsExternal`
- `senderL2Bytes`
- `senderPkts`
- `senderRTO`
- `senderZeroWnd`
- `serverIsExternal`

`receiverBytes:` ***Numéro***

Nombre d'octets au niveau de l'application provenant du récepteur.

`receiverL2Bytes:` ***Numéro***

Le nombre de L2 octets du récepteur.

`receiverPkts:` ***Numéro***

Le nombre de paquets provenant du récepteur.

`receiverRTO:` ***Numéro***

Le nombre de délais de retransmission (RTOS) depuis le récepteur.

`receiverZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par le récepteur.

`roundTripTime:` ***Numéro***

Le temps moyen aller-retour (RTT), exprimé en millisecondes. La valeur est `NaN` s'il n'y a pas d'échantillons RTT.

`senderBytes:` ***Numéro***

Le nombre d'octets au niveau de l'application provenant de l'expéditeur.

`senderL2Bytes:` ***Numéro***

Le nombre de L2 octets de l'expéditeur.

`senderPkts:` ***Numéro***

Le nombre de paquets provenant de l'expéditeur.

`senderRTO:` ***Numéro***

Le nombre de délais de retransmission (RTOS) de l'expéditeur.

`senderZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par l'expéditeur.

TFTP

La classe TFTP (Trivial File Transfer Protocol) vous permet de stocker des métriques et d' accéder à des propriétés sur TFTP_REQUEST et TFTP_RESPONSE événements.

Évènements

TFTP_REQUESTS

S'exécute sur chaque requête TFTP traitée par l'équipement.

TFTP_RESPONSE

S'exécute sur chaque réponse TFTP traitée par l'équipement.

Méthodes

commitRecord(): ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur TFTP_RESPONSE événement. Enregistrez les validations le TFTP_REQUEST les événements ne sont pas pris en charge.

Pour afficher les propriétés par défaut validées pour l'objet d'enregistrement, consultez record propriété ci-dessous.

Pour les enregistrements intégrés, chaque enregistrement unique n'est validé qu'une seule fois, même si commitRecord() La méthode est appelée plusieurs fois pour le même enregistrement unique.

Propriétés

blocks: ***Numéro***

Le nombre de blocs de données écrits ou lus.

Accès uniquement sur TFTP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

error: ***Corde | nul***

Message d'erreur détaillé enregistré par le système ExtraHop.

Accès uniquement sur TFTP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

fileComplete: ***Booléen***

Si la valeur est false, seule une partie du fichier a été transférée, soit parce que le client a expiré pendant une opération d'écriture, soit parce que le serveur a expiré pendant une opération de lecture.

Accès uniquement sur TFTP_RESPONSE événements ; dans le cas contraire, une erreur se produira.

filename: ***Corde***

Le nom du fichier transféré.

mode: ***Corde***

Mode avec lequel le fichier a été transféré. Les valeurs suivantes sont valides :

- netascii
- octet
- mail

operation: ***Corde***

L'opération TFTP. Les valeurs suivantes sont valides :

- READ
- WRITE

`payload:` ***Tampon***

Le Tampon objet qui contient les octets de charge utile bruts du premier bloc de données transféré. La taille maximale d'un bloc est de 512 octets.

`payloadMediaType:` ***Corde***

Type de fichier transféré.

Accès uniquement sur `TFTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`payloadSHA256:` ***Corde***

Représentation hexadécimale du hachage SHA-256 de la charge utile. La chaîne ne contient aucun délimiteur, comme illustré dans l'exemple suivant :

```
468c6c84db844821c9ccb0983c78d1cc05327119b894b5ca1c6a1318784d3675
```

Accès uniquement sur `TFTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`size:` ***Numéro***

Taille du fichier transféré, exprimée en octets.

Accès uniquement sur `TFTP_RESPONSE` événements ; dans le cas contraire, une erreur se produira.

`Turn`

`Turn` est une classe qui vous permet de stocker des métriques et d'accéder aux propriétés disponibles sur `FLOW_TURN` événements.

Le `FLOW_TURN` l'événement est défini dans le Flow section.

Propriétés

`clientBytes:` ***Numéro***

La taille de la demande que le client transféré, exprimé en octets.

`clientTransferTime:` ***Numéro***

Le temps de transfert du client, exprimé en millisecondes.

`processingTime:` ***Numéro***

Temps écoulé entre le moment où le client transfère la demande au serveur et le moment où le serveur commence à renvoyer la réponse au client, exprimé en millisecondes.

`reqSize:` ***Numéro***

Taille de la charge utile de la demande, exprimée en octets.

`reqTransferTime:` ***Numéro***

Le temps de transfert de la demande, exprimé en millisecondes. Si la demande est contenue dans un seul paquet, le temps de transfert est nul. Si la demande couvre plusieurs paquets, la valeur est le délai entre la détection du premier paquet de demande et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une demande importante ou un retard du réseau. La valeur est `NaN` s'il n' y a pas de mesure valide ou si le chronométrage n'est pas valide.

`rspSize:` ***Numéro***

Taille de la charge utile de réponse, exprimée en octets.

`rspTransferTime:` ***Numéro***

Le temps de transfert de réponse, exprimé en millisecondes. Si la réponse est contenue dans un seul paquet, le temps de transfert est nul. Si la réponse couvre plusieurs paquets, la valeur est le délai entre la détection du premier paquet de réponse et la détection du dernier paquet par le système ExtraHop. Une valeur élevée peut indiquer une réponse importante ou un retard du réseau. La valeur est `NaN` s'il n' y a pas de mesure valide ou si le chronométrage n'est pas valide.

`serverBytes:` ***Numéro***

Taille de la réponse transférée par le serveur, exprimée en octets.

`serverTransferTime:` ***Numéro***

Le temps de transfert du serveur, exprimé en millisecondes.

`sourceDevice:` ***Appareil***

L'objet de l'équipement source. Voir le Device cours pour plus d'informations.

`thinkTime:` ***Numéro***

Le temps écoulé entre le transfert de la réponse par le serveur au client et le client transférant une nouvelle demande au serveur, exprimée en millisecondes. La valeur est `NaN` s'il n'y a pas de mesure valide.

UDP

Le `UDP` la classe vous permet d'accéder aux propriétés et de récupérer des métriques à partir d'événements UDP et de `FLOW_TICK` et `FLOW_TURN` événements.

Le `FLOW_TICK` et `FLOW_TURN` les événements sont définis dans le Flow section.

Évènements

`UDP_PAYLOAD`

S'exécute lorsque la charge utile correspond aux critères configurés dans le déclencheur associé .

En fonction du Flow , la charge utile UDP se trouve dans les propriétés suivantes :

- `Flow.client.payload`
- `Flow.payload1`
- `Flow.payload2`
- `Flow.receiver.payload`
- `Flow.sender.payload`
- `Flow.server.payload`

Des options de charge utile supplémentaires sont disponibles lorsque vous créez un déclencheur qui s'exécute lors de cet événement. Voir Options de déclencheur avancées pour plus d' informations.

WebSocket

Le `WebSocket` la classe vous permet d'accéder aux propriétés sur `WEBSOCKET_OPEN`, `WEBSOCKET_CLOSE`, et `WEBSOCKET_MESSAGE` événements.

Évènements

`WEBSOCKET_OPEN`

S'exécute lorsqu'une liaison réussie a été observée dans l'établissement.

`WEBSOCKET_CLOSE`

S'exécute lorsque les deux trames fermées sont observées ou lorsque la connexion TCP sous-jacente est fermée.

`WEBSOCKET_MESSAGE`

S'exécute lorsque tous les cadres d'un texte ou d'un message binaire ont été respectés.

Propriétés

`clientBytes:` ***Numéro***

Le nombre total d'octets envoyés par le client pendant la session WebSockets.

Accès uniquement sur WEBSOCKET_MESSAGE événements ; sinon, une erreur se produira.

clientL2Bytes: ***Numéro***

Le nombre total de L2 octets envoyés par le client pendant la session WebSockets.

Accès uniquement sur WEBSOCKET_MESSAGE événements ; sinon, une erreur se produira.

clientPkts: ***Numéro***

Le nombre total de paquets envoyés par le client pendant la session WebSockets.

Accès uniquement sur WEBSOCKET_MESSAGE événements ; sinon, une erreur se produira.

clientRTO: ***Numéro***

Le nombre total de clients délais de retransmission (RTO) observés lors de la session WebSockets.

Accès uniquement sur WEBSOCKET_MESSAGE événements ; sinon, une erreur se produira.

clientZeroWnd: ***Numéro***

Nombre de fenêtres nulles envoyées par le client.

Accès uniquement sur WEBSOCKET_MESSAGE événements ; sinon, une erreur se produira.

closeReason: ***Corde***

Le message texte inclus dans le premier cadre de fermeture observé qui décrit la raison pour laquelle la connexion a été fermée. La valeur est null si le cadre ne contient pas ces informations.

Accès uniquement sur WEBSOCKET_CLOSE événements ; sinon, une erreur se produira.

host: ***Corde***

L'hôte indiqué dans la demande de liaison faite par le client dans la demande de liaison de l'établissement. La valeur est null si aucun hôte n'est fourni.

Accès uniquement sur WEBSOCKET_OPEN événements ; sinon, une erreur se produira.

isClientClose: ***Booléen***

La valeur est true si le cadre de fermeture initial a été envoyé par le client.

Accès uniquement sur WEBSOCKET_CLOSE événements ; sinon, une erreur se produira.

isEncrypted: ***Booléen***

La valeur est true si la connexion WebSocket est cryptée par TLS.

isMasked: Boolean

La valeur est vraie si les cadres du message WebSocket sont masqués.

Accès uniquement sur WEBSOCKET_MESSAGE événements ; sinon, une erreur se produira.

isServerClose: ***Booléen***

La valeur est true si la trame de fermeture initiale a été envoyée par le serveur. La valeur est false si la connexion s'est interrompue de manière anormale.

Accès uniquement sur WEBSOCKET_CLOSE événements ; sinon, une erreur se produira.

msg: ***Tampon***

Le Tampon objet contenant le message WebSocket. Si le message est compressé, le tampon contient le message décompressé. Le tampon est null si le contenu dépasse la longueur maximale.

Accès uniquement sur WEBSOCKET_MESSAGE événements ; sinon, une erreur se produira.

msgLength: ***Numéro***

Longueur du message, exprimée en octets. Si le message est compressé, la longueur reflète la longueur totale du message décompressé, même si le message dépasse la longueur maximale.

Accès uniquement sur WEBSOCKET_MESSAGE événements ; sinon, une erreur se produira.

msgType: ***Corde***

Type de cadre de message WebSocket. Les valeurs valides sont TEXT ou BINARY.

Accès uniquement sur WEBSOCKET_MESSAGE événements ; sinon, une erreur se produira.

`origin:` ***Corde***

URL d'origine fournie dans la demande de liaison initiée par le client pour l'établissement dgociée.

Accès uniquement sur `WEBSOCKET_OPEN` événements ; sinon, une erreur se produira.

`rawMsgLength:` ***Numéro***

Longueur du message brut tel qu'il a été observé, exprimée en octets. Si le message est compressé, cette propriété reflète la longueur du message compressé.

Accès uniquement sur `WEBSOCKET_MESSAGE` événements ; sinon, une erreur se produira.

`serverBytes:` ***Numéro***

Nombre total d'octets renvoyés par le serveur pendant la session WebSockets.

Accès uniquement sur `WEBSOCKET_MESSAGE` événements ; sinon, une erreur se produira.

`serverL2Bytes:` ***Numéro***

Le nombre total de L2 octets renvoyés par le serveur lors de la session WebSockets.

Accès uniquement sur `WEBSOCKET_MESSAGE` événements ; sinon, une erreur se produira.

`serverPkts:` ***Numéro***

Le nombre total de paquets renvoyés par le serveur pendant la session WebSockets.

Accès uniquement sur `WEBSOCKET_MESSAGE` événements ; sinon, une erreur se produira.

`serverRTO:` ***Numéro***

Le nombre total de serveurs délais de retransmission (RTO) observés lors de la session WebSockets.

Accès uniquement sur `WEBSOCKET_MESSAGE` événements ; sinon, une erreur se produira.

`serverZeroWnd:` ***Numéro***

Le nombre de fenêtres nulles envoyées par le serveur.

Accès uniquement sur `WEBSOCKET_MESSAGE` événements ; sinon, une erreur se produira.

`statusCode:` ***Numéro***

Le code dquo qui représente la raison pour laquelle la connexion a été fermée, tel que défini dans la RFC 6455.

La valeur est `NO_STATUS_RECVD` (`1005`) si le cadre de fermeture initial ne comporte pas de code dquo. La valeur est `NaN` si la connexion s'est interrompue de manière anormale.

Accès uniquement sur `WEBSOCKET_CLOSE` événements ; sinon, une erreur se produira.

`uri:` ***Corde***

L'URI fourni dans la demande de liaison initiée par le client pour l'établissement dicible.

Accès uniquement sur `WEBSOCKET_OPEN` événements ; sinon, une erreur se produira.

`WSMAN`

Le `WSMAN` la classe vous permet de stocker des métriques et d'accéder aux propriétés sur `WSMAN_REQUEST` et `WSMAN_RESPONSE` événements. Web Services-Management (WSMAN) et l'implémentation Microsoft de Windows Remote Management (WinRM) sont des protocoles qui permettent aux appareils d'échanger des informations de gestion sur un réseau.

Évènements

`WSMAN_REQUEST`

Fonctionne sur tous `WSMAN_REQUEST` traité par l'équipement.

`WSMAN_RESPONSE`

Fonctionne sur tous `WSMAN_RESPONSE` traité par l'équipement.

Méthodes

`commitRecord()`: ***vide***

Envoie un enregistrement à l'espace de stockage des enregistrements configuré sur un `WSMAN_REQUEST` ou `WSMAN_RESPONSE` événement. Pour afficher les propriétés par défaut validées pour chaque événement, consultez la propriété d'enregistrement ci-dessous.

Si le `commitRecord()` la méthode est appelée sur un `WSMAN_REQUEST` événement, l'enregistrement n'est pas créé avant le `WSMAN_RESPONSE` l'événement se déroule. Si le `commitRecord()` la méthode est appelée à la fois sur `WSMAN_REQUEST` et le correspondant `WSMAN_RESPONSE`, un seul enregistrement est créé pour la demande et la réponse, même si `commitRecord()` la méthode est appelée plusieurs fois sur les mêmes événements déclencheurs.

Propriétés

`encryptionProtocol`: ***Corde***

Le protocole avec lequel la transaction est cryptée.

`isEncrypted`: ***Booléen***

La valeur est `true` si la transaction est effectuée via le protocole HTTP sécurisé.

`isDecrypted`: ***Booléen***

La valeur est `true` si le système ExtraHop a déchiffré et analysé la transaction en toute sécurité. L'analyse du trafic déchiffré peut révéler les menaces avancées qui se cachent dans le trafic chiffré.

`operationId`: ***Corde***

Identifiant unique de l'opération.

`payload`: ***Tampon***

Un objet tampon contenant l'enveloppe du message XML. Les messages dont la taille est supérieure à la taille maximale sont tronqués. La taille maximale est configurée dans le profil WSMAN de la configuration en cours. L'exemple de configuration en cours d'exécution suivant modifie la taille maximale des messages de 1024 octets par défaut à 4096 :

```
"capture": {
    "app_proto": {
        "wsman": {
            "payload_max_size": 4096
        }
    }
}
```

`record`: ***Objet***

L'objet d'enregistrement qui peut être envoyé à l'espace de stockage des enregistrements configuré via un appel à `WSMAN.commitRecord()`.

L'objet d'enregistrement par défaut peut contenir les propriétés suivantes :

- `clientAddr`
- `clientIsExternal`
- `clientPort`
- `serverAddr`
- `serverPort`
- `proto`
- `timestamp`
- `user`
- `vlan`
- `operationId`
- `receiverIsExternal`
- `reqAction`

- reqResourceURI
- rspAction
- rspResourceURI
- senderIsExternal
- sequenceId
- serverIsExternal

Accédez à l'objet d'enregistrement uniquement sur WSMAN_RESPONSE événements ; sinon, une erreur se produira.

reqAction: ***Corde***

Action demandée par le client à exécuter par la ressource spécifiée dans le ResourceURI.

Accès uniquement sur WSMAN_REQUEST événements ; sinon, une erreur se produira.

reqCommand: ***Corde | nul***

La commande spécifiée dans la demande. Si aucune commande n'est spécifiée, la valeur est nulle.

reqResourceURI: ***Corde***

L'identifiant de ressource uniforme (URI) de la ressource qui exécute une action.

rspAction: ***Corde***

La réponse du serveur à l'action demandée par le client.

Accès uniquement sur WSMAN_RESPONSE événements ; sinon, une erreur se produira.

rspResourceURI: ***Corde***

L'identifiant de ressource uniforme (URI) de la ressource qui exécute une action.

sequenceId: ***Corde***

Représentation sous forme de chaîne d'un entier de 64 bits identifiant un message dans le cadre d'une opération.

user: ***Corde***

Le nom d'utilisateur du compte qui a envoyé la demande.

Classes de flux de données ouvertes

Les classes d'API Trigger présentées dans cette section vous permettent d'envoyer des données à un syslog, à une base de données ou à un serveur tiers via un flux de données ouvert (ODS) que vous avez configuré dans les paramètres d'administration.

Classe	Descriptif
Remote.HTTP	Vous permet de soumettre des données de requête HTTP à un serveur distant via les points de terminaison de l'API REST.
Remote.Kafka	Permet d'envoyer des données de message à un serveur Kafka distant.
Remote.MongoDB	Vous permet d'insérer, de supprimer et de mettre à jour des collections de documents sur une télécommande MongoDB base de données.
Remote.Raw	Vous permet de soumettre des données brutes à un serveur distant via un port TCP ou UDP.
Remote.Syslog	Vous permet d'envoyer des données Syslog à un serveur distant.

`Remote.HTTP`

Le `Remote.HTTP` le cours vous permet de soumettre HTTP demander des données à un HTTP flux de données ouvert (ODS) cible et fournit un accès aux points de terminaison de l'API HTTP REST.

Vous devez d'abord configurer une cible HTTP ODS à partir des paramètres d'administration, ce qui nécessite des privilèges d'administration du système et des accès. Pour plus d'informations sur la configuration, consultez Flux de données ouverts section du Guide d'administration d'ExtraHop.

Méthodes

`delete`

Soumet une demande de suppression HTTP REST à un flux de données ouvert HTTP configuré.

Syntaxe :

```
Remote.HTTP("name").delete({path: "path", headers: {header:
 "header"},
payload: "payload"})
```

```
Remote.HTTP.delete({path: "path", headers: {header: "header"},
 payload: "payload"})
```

Paramètres :

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`options:` ***Objet***

L'objet options possède les propriétés suivantes :

`path:` ***Corde***

Chaîne spécifiant le chemin de la demande.

`headers:` ***Objet***

L'objet facultatif spécifiant les en-têtes de demande. Les en-têtes suivants sont restreints et provoqueront une erreur s'ils sont spécifiés :

- `Connection`
- `Authorization`
- `Proxy-Connection`
- `Content-Length`
- `X-Forwarded-For`
- `Transfer-Encoding`

Note: Les en-têtes d'autorisation doivent être spécifiés soit par une méthode d'authentification intégrée, telle qu'Amazon Web Services, soit par le biais du **En-tête HTTP supplémentaire** champ dans le Flux de données ouverts fenêtre de configuration dans les paramètres d'administration.

Les en-têtes configurés dans un déclencheur ont priorité sur une entrée du **En-tête HTTP supplémentaire** champ, qui se trouve dans Flux de données ouverts fenêtre de configuration dans les paramètres d'administration. Par exemple, si **En-tête HTTP supplémentaire** le champ spécifie `Content-Type: text/plain`, mais un script déclencheur sur la même cible ODS spécifie `Content-Type: application/json`, puis `Content-Type: application/json` est inclus dans la requête HTTP.

Vous pouvez compresser les requêtes HTTP sortantes avec l' en-tête Content-Encoding.

```
'Content-Encoding': 'gzip'
```

Les valeurs suivantes sont prises en charge pour cet en-tête de compression :

- `gzip`
- `deflate`

`payload:` ***Corde*** | ***Tampon***

La chaîne ou le tampon facultatif spécifiant la charge utile de la demande.

Valeurs renvoyées :

Retours `true` si la demande est en file d'attente, sinon renvoie `false`.

`get`

Soumet une requête HTTP REST get à un flux de données ouvert HTTP configuré.

Syntaxe :

```
Remote.HTTP("name").get({path: "path", headers: {header:
 "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.get({path: "path", headers: {header: "header"},
 payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

Paramètres :

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`options:` ***Objet***

L'objet options possède les propriétés suivantes :

`path:` ***Corde***

Chaîne spécifiant le chemin de la demande.

`headers:` ***Objet***

L'objet facultatif spécifiant les en-têtes de demande. Les en-têtes suivants sont restreints et provoqueront une erreur s'ils sont spécifiés :

- `Connection`
- `Authorization`
- `Proxy-Connection`
- `Content-Length`
- `X-Forwarded-For`
- `Transfer-Encoding`

Note: Les en-têtes d'autorisation doivent être spécifiés soit par une méthode d'authentification intégrée, telle qu'Amazon Web Services, soit par le biais du **En-tête HTTP supplémentaire** champ dans le Flux de données ouverts fenêtre de configuration dans les paramètres d'administration.

Les en-têtes configurés dans un déclencheur ont priorité sur une entrée du **En-tête HTTP supplémentaire** champ, qui se trouve dans Flux de données ouverts fenêtre de configuration dans les paramètres d'administration. Par exemple, si **En-tête HTTP supplémentaire** le champ spécifie `Content-Type: text/plain`, mais un script déclencheur sur la même cible ODS spécifie `Content-Type: application/json`, puis `Content-Type: application/json` est inclus dans la requête HTTP.

Vous pouvez compresser les requêtes HTTP sortantes avec l' en-tête Content-Encoding.

```
'Content-Encoding': 'gzip'
```

Les valeurs suivantes sont prises en charge pour cet en-tête de compression :

- `gzip`
- `deflate`

`payload:` ***Corde*** | ***Tampon***

La chaîne ou le tampon facultatif spécifiant la charge utile de la demande.

`enableResponseEvent:` ***Booléen***

Permet à un déclencheur de s'exécuter sur la Réponse HTTP envoyée par la cible ODS en créant un événement REMOTE_RESPONSE.

Important: Le traitement d'un grand nombre de réponses HTTP peut affecter les performances et l'efficacité du déclencheur. Nous vous recommandons d'activer cette option uniquement si cela est nécessaire.

`context:` ***Objet*** | ***Corde*** | ***Numéro*** | ***Booléen*** | ***nul***

Objet facultatif envoyé au déclencheur qui s'exécute sur la Réponse HTTP de la cible ODS. Vous pouvez accéder aux informations stockées dans l'objet en spécifiant le `Remote.response.context` propriété.

Valeurs renvoyées :

Retours `true` si la demande est en file d'attente, sinon renvoie `false`.

`patch`

Soumet une demande de correctif HTTP REST à un flux de données ouvert HTTP configuré.

Syntaxe :

```
Remote.HTTP("name").patch({path: "path", headers: {header:
 "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.patch({path: "path", headers: {header: "header"},
 payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

Paramètres :

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`options:` ***Objet***

L'objet options possède les propriétés suivantes :

`path:` ***Corde***

Chaîne spécifiant le chemin de la demande.

`headers:` ***Objet***

L'objet facultatif spécifiant les en-têtes de demande. Les en-têtes suivants sont restreints et provoqueront une erreur s'ils sont spécifiés :

- `Connection`
- `Authorization`
- `Proxy-Connection`
- `Content-Length`
- `X-Forwarded-For`
- `Transfer-Encoding`

Note: Les en-têtes d'autorisation doivent être spécifiés soit par une méthode d'authentification intégrée, telle qu'Amazon Web Services, soit par le biais du **En-tête HTTP supplémentaire** champ dans le Flux de données ouverts fenêtre de configuration dans les paramètres d'administration.

Les en-têtes configurés dans un déclencheur ont priorité sur une entrée du **En-tête HTTP supplémentaire** champ, qui se trouve dans Flux de données ouverts fenêtre de configuration dans les paramètres d'administration. Par exemple, si **En-tête HTTP supplémentaire** le champ spécifie `Content-Type: text/plain`, mais un script déclencheur sur la même cible ODS spécifie `Content-Type: application/json`, puis `Content-Type: application/json` est inclus dans la requête HTTP.

Vous pouvez compresser les requêtes HTTP sortantes avec l' en-tête Content-Encoding.

```
'Content-Encoding': 'gzip'
```

Les valeurs suivantes sont prises en charge pour cet en-tête de compression :

- `gzip`
- `deflate`

`payload:` ***Corde*** | ***Tampon***

La chaîne ou le tampon facultatif spécifiant la charge utile de la demande.

`enableResponseEvent:` ***Booléen***

Permet à un déclencheur de s'exécuter sur la Réponse HTTP envoyée par la cible ODS en créant un événement REMOTE_RESPONSE.

Important: Le traitement d'un grand nombre de réponses HTTP peut affecter les performances et l'efficacité du déclencheur. Nous vous recommandons d'activer cette option uniquement si cela est nécessaire.

`context:` ***Objet*** | ***Corde*** | ***Numéro*** | ***Booléen*** | ***nul***

Objet facultatif envoyé au déclencheur qui s'exécute sur la Réponse HTTP de la cible ODS. Vous pouvez accéder aux informations stockées dans l'objet en spécifiant le `Remote.response.context` propriété.

Valeurs renvoyées :

Retours `true` si la demande est en file d'attente, sinon renvoie `false`.

`post`

Soumet une requête de publication HTTP REST à un flux de données ouvert HTTP configuré.

Syntaxe :

```
Remote.HTTP("name").post({path: "path", headers: {header:
 "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.post({path: "path", headers: {header: "header"},
 payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

Paramètres :

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`options:` ***Objet***

L'objet options possède les propriétés suivantes :

`path:` ***Corde***

Chaîne spécifiant le chemin de la demande.

`headers:` ***Objet***

L'objet facultatif spécifiant les en-têtes de demande. Les en-têtes suivants sont restreints et provoqueront une erreur s'ils sont spécifiés :

- `Connection`
- `Authorization`
- `Proxy-Connection`

- `Content-Length`
- `X-Forwarded-For`
- `Transfer-Encoding`

Note: Les en-têtes d'autorisation doivent être spécifiés soit par une méthode d'authentification intégrée, telle qu'Amazon Web Services, soit par le biais du **En-tête HTTP supplémentaire** champ dans le Flux de données ouverts fenêtre de configuration dans les paramètres d'administration.

Les en-têtes configurés dans un déclencheur ont priorité sur une entrée du **En-tête HTTP supplémentaire** champ, qui se trouve dans Flux de données ouverts fenêtre de configuration dans les paramètres d'administration. Par exemple, si **En-tête HTTP supplémentaire** le champ spécifie `Content-Type: text/plain`, mais un script déclencheur sur la même cible ODS spécifie `Content-Type: application/json`, puis `Content-Type: application/json` est inclus dans la requête HTTP.

Vous pouvez compresser les requêtes HTTP sortantes avec l' en-tête Content-Encoding.

```
'Content-Encoding': 'gzip'
```

Les valeurs suivantes sont prises en charge pour cet en-tête de compression :

- `gzip`
- `deflate`

`payload:` ***Corde | Tampon***

La chaîne ou le tampon facultatif spécifiant la charge utile de la demande.

`enableResponseEvent:` ***Booléen***

Permet à un déclencheur de s'exécuter sur la Réponse HTTP envoyée par la cible ODS en créant un événement REMOTE_RESPONSE.

Important: Le traitement d'un grand nombre de réponses HTTP peut affecter les performances et l'efficacité du déclencheur. Nous vous recommandons d'activer cette option uniquement si cela est nécessaire.

`context:` ***Objet*** | ***Corde*** | ***Numéro*** | ***Booléen*** | ***nul***

Objet facultatif envoyé au déclencheur qui s'exécute sur la Réponse HTTP de la cible ODS. Vous pouvez accéder aux informations stockées dans l'objet en spécifiant le `Remote.response.context` propriété.

Valeurs renvoyées :

Retours `true` si la demande est en file d'attente, sinon renvoie `false`.

`put`

Soumet une requête HTTP REST put à un flux de données ouvert HTTP configuré.

Syntaxe :

```
Remote.HTTP("name").put({path: "path", headers: {header:
 "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.put({path: "path", headers: {header: "header"},
 payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

Paramètres :

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`options:` ***Objet***

L'objet options possède les propriétés suivantes :

`path:` ***Corde***

Chaîne spécifiant le chemin de la demande.

`headers:` ***Objet***

L'objet facultatif spécifiant les en-têtes de demande. Les en-têtes suivants sont restreints et provoqueront une erreur s'ils sont spécifiés :

- `Connection`
- `Authorization`
- `Proxy-Connection`
- `Content-Length`
- `X-Forwarded-For`
- `Transfer-Encoding`

Note: Les en-têtes d'autorisation doivent être spécifiés soit par une méthode d'authentification intégrée, telle qu'Amazon Web Services, soit par le biais du **En-tête HTTP supplémentaire** champ dans le Flux de données ouverts fenêtre de configuration dans les paramètres d'administration.

Les en-têtes configurés dans un déclencheur ont priorité sur une entrée du **En-tête HTTP supplémentaire** champ, qui se trouve dans Flux de données ouverts fenêtre de configuration dans les paramètres d'administration. Par exemple, si **En-tête HTTP supplémentaire** le champ spécifie `Content-Type: text/plain`, mais un script déclencheur sur la même cible ODS spécifie `Content-Type: application/json`, puis `Content-Type: application/json` est inclus dans la requête HTTP.

Vous pouvez compresser les requêtes HTTP sortantes avec l' en-tête Content-Encoding.

```
'Content-Encoding': 'gzip'
```

Les valeurs suivantes sont prises en charge pour cet en-tête de compression :

- `gzip`
- `deflate`

`payload:` ***Corde*** | ***Tampon***

La chaîne ou le tampon facultatif spécifiant la charge utile de la demande.

`enableResponseEvent:` ***Booléen***

Permet à un déclencheur de s'exécuter sur la Réponse HTTP envoyée par la cible ODS en créant un événement REMOTE_RESPONSE.

Important: Le traitement d'un grand nombre de réponses HTTP peut affecter les performances et l'efficacité du déclencheur. Nous vous recommandons d'activer cette option uniquement si cela est nécessaire.

`context:` ***Objet*** | ***Corde*** | ***Numéro*** | ***Booléen*** | ***nul***

Objet facultatif envoyé au déclencheur qui s'exécute sur la Réponse HTTP de la cible ODS. Vous pouvez accéder aux informations stockées dans l'objet en spécifiant le `Remote.response.context` propriété.

Valeurs renvoyées :

Retours `true` si la demande est en file d'attente, sinon renvoie `false`.

`request`

Soumet une requête HTTP REST à un flux de données ouvert HTTP configuré.

Syntaxe :

```
Remote.HTTP("name").request("method", {path: "path", headers:
 {header: "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.request("method", {path: "path", headers: {header:
 "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

Paramètres :

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`method:` ***Corde***

Chaîne qui spécifie la méthode HTTP.

- `GET`
- `HEAD`
- `POST`
- `PUT`
- `DELETE`
- `TRACE`
- `OPTIONS`
- `CONNECT`
- `PATCH`

`options:` ***Objet***

L'objet options possède les propriétés suivantes :

`path:` ***Corde***

Chaîne spécifiant le chemin de la demande.

`headers:` ***Objet***

L'objet facultatif spécifiant les en-têtes de demande. Les en-têtes suivants sont restreints et provoqueront une erreur s'ils sont spécifiés :

- `Connection`
- `Authorization`
- `Proxy-Connection`
- `Content-Length`
- `X-Forwarded-For`
- `Transfer-Encoding`

Note: Les en-têtes d'autorisation doivent être spécifiés soit par une méthode d'authentification intégrée, telle qu'Amazon Web Services, soit par le biais du **En-tête HTTP supplémentaire** champ dans le Flux de données ouverts fenêtre de configuration dans les paramètres d'administration.

Les en-têtes configurés dans un déclencheur ont priorité sur une entrée du **En-tête HTTP supplémentaire** champ, qui se trouve dans Flux de données ouverts fenêtre de configuration dans les paramètres d'administration. Par exemple, si **En-tête HTTP supplémentaire** le champ spécifie `Content-Type: text/plain`, mais un script déclencheur sur la même cible ODS spécifie `Content-Type: application/json`, puis `Content-Type: application/json` est inclus dans la requête HTTP.

Vous pouvez compresser les requêtes HTTP sortantes avec l' en-tête Content-Encoding.

```
'Content-Encoding': 'gzip'
```

Les valeurs suivantes sont prises en charge pour cet en-tête de compression :

- `gzip`
- `deflate`

`payload:` ***Corde*** | ***Tampon***

La chaîne ou le tampon facultatif spécifiant la charge utile de la demande.

`enableResponseEvent:` ***Booléen***

Permet à un déclencheur de s'exécuter sur la Réponse HTTP envoyée par la cible ODS en créant un événement REMOTE_RESPONSE.

Important: Le traitement d'un grand nombre de réponses HTTP peut affecter les performances et l'efficacité du déclencheur. Nous vous recommandons d'activer cette option uniquement si cela est nécessaire.

`context:` ***Objet*** | ***Corde*** | ***Numéro*** | ***Booléen*** | ***nul***

Objet facultatif envoyé au déclencheur qui s'exécute sur la Réponse HTTP de la cible ODS. Vous pouvez accéder aux informations stockées dans l'objet en spécifiant le `Remote.response.context` propriété.

Valeurs renvoyées :

Retours `true` si la demande est en file d'attente, sinon renvoie `false`.

Méthodes auxiliaires

Les méthodes d'assistance suivantes sont disponibles pour les méthodes HTTP courantes.

- `Remote.HTTP.delete`
- `Remote.HTTP.get`
- `Remote.HTTP.patch`
- `Remote.HTTP.post`
- `Remote.HTTP.put`

Syntaxe :

```
Remote.HTTP("name").delete({path: "path", headers: {header: "header"},
```

```
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.delete({path: "path", headers: {header: "header"}, payload:
 "payload", enableResponseEvent: "enableResponseEvent", context:
 "context"})
```

```
Remote.HTTP("name").get({path: "path", headers: {header: "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.get({path: "path", headers: {header: "header"}, payload:
 "payload", enableResponseEvent: "enableResponseEvent", context:
 "context"})
```

```
Remote.HTTP("name").patch({path: "path", headers: {header: "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.patch({path: "path", headers: {header: "header"}, payload:
 "payload", enableResponseEvent: "enableResponseEvent", context:
 "context"})
```

```
Remote.HTTP("name").post({path: "path", headers: {header: "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.post({path: "path", headers: {header: "header"}, payload:
 "payload", enableResponseEvent: "enableResponseEvent", context:
 "context"})
```

```
Remote.HTTP("name").put({path: "path", headers: {header: "header"},
payload: "payload", enableResponseEvent: "enableResponseEvent",
 context: "context"})
```

```
Remote.HTTP.put({path: "path", headers: {header: "header"}, payload:
 "payload", enableResponseEvent: "enableResponseEvent", context:
 "context"})
```

Valeurs renvoyées :

Retours `true` si la demande est en file d'attente, sinon renvoie `false`.

Exemples

`HTTP GET`

L'exemple suivant enverra une requête HTTP GET à la configuration HTTP appelée « my_destination » et un chemin correspondant à l'URI, y compris les variables de chaîne de requête, à laquelle vous souhaitez que la demande soit envoyée .

```
Remote.HTTP("my_destination").get( { path: "/?
example=example1&example2=my_data" } );
```

`HTTP POST`

L'exemple suivant enverra une requête HTTP POST à la configuration HTTP appelée « my_destination », le chemin correspondant à l'URI à laquelle vous souhaitez que la demande soit

envoyée et une charge utile. La charge utile peut être constituée de données similaires à celles d'un HTTP client enverrait, un blob JSON, du XML ou tout ce que vous souhaitez envoyer.

```
Remote.HTTP("my_destination").post( { path: "/", payload: "data I want
 to
send" } );
```

En-têtes HTTP personnalisés

L'exemple suivant définit un objet Javascript avec des clés pour représenter les noms des en-têtes et leurs valeurs correspondantes et les fournir dans un appel en tant que valeur de la clé des en-têtes.

```
var my_json = { example: "my_data", example1: 42, example2: false };
var headers = { "Content-Type": "application/json" };
Remote.HTTP("my_destination").post( { path: "/", headers: headers,
 payload:
JSON.stringify(my_json) });
```

Exemples de déclencheurs

- Exemple : envoyer des données à Elasticsearch avec Remote.http
- Exemple : envoyer des données à Azure avec Remote.http

`Remote.Kafka`

Le `Remote.Kafka` classe vous permet de soumettre des données de message à un serveur Kafka via un Kafka flux de données ouvert (ODS).

Vous devez d'abord configurer une cible Kafka ODS depuis les paramètres d'administration, ce qui nécessite des privilèges d'administration du système et des accès. Pour obtenir des informations de configuration, consultez le Flux de données ouverts section du Guide de l'interface utilisateur ExtraHop Admin.

Méthodes

`send`

Envoie un ensemble de messages à un seul sujet avec une option permettant d'indiquer à quelle partition Kafka les messages seront envoyés.

Syntaxe :

```
Remote.Kafka.send({"topic": "topic", "messages":[messages],
"partition": partition})
```

```
Remote.Kafka("name").send({"topic": "topic", "messages":
[messages],
"partition": partition})
```

Paramètres :

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`topic:` ***Corde***

Une chaîne correspondant au sujet associé au Kafka `send` méthode. La chaîne de rubrique comporte les restrictions suivantes :

- La longueur de la chaîne doit être comprise entre 1 et 249 caractères.

- La chaîne ne prend en charge que les caractères alphanumériques et les symboles suivants : « - », « _ » ou «. ».
- La chaîne ne peut pas être «. » ou «.. ».

`messages:` ***Array***

Tableau facultatif de messages à envoyer. Un élément de ce tableau ne peut pas être un tableau lui-même.

`partition:` ***Numéro***

Un entier non négatif facultatif correspondant à la partition Kafka à laquelle les messages seront envoyés. Le `send` l'action échouera silencieusement si le nombre fourni dépasse le nombre de partitions du cluster Kafka associé à la cible donnée. Cette valeur est ignorée sauf si **Partitionnement manuel** est sélectionnée comme stratégie de partitionnement lorsque vous avez configuré le flux de données ouvert dans les paramètres d'administration.

Valeurs renvoyées :

Aucune

Exemples :

```
Remote.Kafka.send({"topic": "my_topic", "messages": ["hello
 world", 42,
DHCP.msgType], "partition": 2});
```

```
Remote.Kafka("my-target").send({"topic": "my_topic", "messages":
 [HTTP.query,
HTTP.uri]});
```

`send`

Envoie des messages à un seul sujet.

Syntaxe :

```
Remote.Kafka.send("topic", message1, message2, etc...)
```

```
Remote.Kafka("my-target").send("topic", message1, message2,
 etc...)
```

Paramètres :

Si `Remote.Kafka.send` est appelé avec plusieurs arguments, les champs suivants sont obligatoires :

`topic:` ***Corde***

Une chaîne correspondant au sujet associé au Kafka `send` méthode. La chaîne de rubrique comporte les restrictions suivantes :

- La longueur de la chaîne doit être comprise entre 1 et 249 caractères.
- La chaîne ne prend en charge que les caractères alphanumériques et les symboles suivants : « - », « _ » ou «. ».
- La chaîne ne peut pas être «. » ou «.. ».

`messages:` ***Corde | Numéro***

Les messages à envoyer. Il ne peut pas s'agir d'un tableau.

Valeurs renvoyées :

Aucune.

Exemples :

```
Remote.Kafka.send("my_topic", HTTP.query, HTTP.uri);
```

```
Remote.Kafka("my-target").send("my_topic", HTTP.query, HTTP.uri);
```

Remote.MongoDB

Le Remote.MongoDB la classe vous permet d'insérer, de supprimer et de mettre à jour MongoDB collections de documents via une MongoDB flux de données ouvert (ODS).

Vous devez d'abord configurer une cible MongoDB ODS à partir des paramètres d'administration, ce qui nécessite des privilèges d'administration du système et des accès. Pour obtenir des informations de configuration, consultez le Flux de données ouverts section du Guide de l'interface utilisateur ExtraHop Admin.

Méthodes

insert

Insère un document ou un ensemble de documents dans une collection et gère les opérations d'ajout et de modification.

Syntaxe :

```
Remote.MongoDB.insert("db.collection", document);
```

```
Remote.MongoDB("name").insert("db.collection", document);
```

Paramètres :

name : ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur default.

collection : ***Corde***

Le nom d'un groupe de documents MongoDB.

document : ***Objet***

Document au format JSON à insérer dans la collection.

Valeurs renvoyées :

Retours true si la demande est en file d'attente, sinon renvoie false.

Exemples :

```
Remote.MongoDB.insert('sessions.sess_www',
   {
   'session_id': "100",
   'path': "/index.html",
   'host': "www.extrahop.com",
   'status': "500",
   'src_ip': "10.10.1.120",
   'dst_ip': "10.10.1.100"
   }
);
var x = Remote.MongoDB.insert('test.tbc', {example: 1});
if (x) {
   Network.metricAddCount('perf_trigger_success', 1);
}
else {
   Network.metricAddCount('perf_trigger_error', 1);
```

```
}
```

Référez-vous à http://docs.mongodb.org/manual/reference/method/db.collection.insert/#db.collection.insert pour plus d'informations.

`remove`

Supprime des documents d'une collection.

Syntaxe :

```
Remote.MongoDB.remove("collection", document, justOnce);
```

```
Remote.MongoDB("name").remove("collection", document, justOnce]);
```

Paramètres :

`name:` ***Corde***

Le nom facultatif de l'hôte spécifié lorsque vous avez configuré le flux de données ouvert dans les paramètres d'administration. Si aucun hôte n'est spécifié, la valeur est l'hôte par défaut.

`collection:` ***Corde***

Le nom d'un groupe de documents MongoDB.

`document:` ***Objet***

Document au format JSON à supprimer de la collection.

`justOnce:` ***Booléen***

Paramètre booléen facultatif qui limite la suppression à un seul document. Régler sur `true` pour limiter la suppression. La valeur par défaut est `false`.

Valeurs renvoyées :

Retours `true` si la demande est en file d'attente, sinon renvoie `false`.

Exemples :

```
var x = Remote.MongoDB.remove('test.tbc', {qty: 100000}, false);
if (x) {
   Network.metricAddCount('perf_trigger_success', 1);
}
else {
   Network.metricAddCount('perf_trigger_error', 1);
}
```

Référez-vous à http://docs.mongodb.org/manual/reference/method/db.collection.remove/#db.collection.remove pour plus d'informations.

`update`

Modifie un ou plusieurs documents existants d'une collection.

Syntaxe :

```
Remote.MongoDB.update("collection", document, update,
 {"upsert":true,
"multi":true});
```

```
Remote.MongoDB("name").update("collection", document, update,
{"upsert":true, "multi":true});
```

Paramètres :

`collection:` ***Corde***

Le nom d'un groupe de documents MongoDB.

`document:` ***Objet***

Document au format JSON qui indique les documents à mettre à jour ou à insérer, si l'option upsert est définie sur true.

`update:` ***Objet***

Document au format JSON qui indique comment mettre à jour les documents spécifiés.

`name:` ***Corde***

Le nom de l'hôte spécifié lorsque vous avez configuré le flux de données ouvert dans les paramètres d'administration. Si aucun hôte n'a été spécifié, la valeur est l'hôte par défaut.

`options:`

Indicateurs facultatifs indiquant les options de mise à jour supplémentaires suivantes :

`upsert:` ***Booléen***

Paramètre booléen facultatif qui crée un nouveau document lorsqu'aucun document ne correspond aux données de la requête. Régler sur `true` pour créer un nouveau document. La valeur par défaut est `false`.

`multi:` ***Booléen***

Paramètre booléen facultatif qui met à jour tous les documents correspondant aux données de la requête. Régler sur `true` pour mettre à jour plusieurs documents. La valeur par défaut est `false`, qui met à jour uniquement le premier document renvoyé.

Valeurs renvoyées :

La valeur est `true` si la demande est en file d'attente, sinon renvoie FALSE.

Exemples :

```
var x = Remote.MongoDB.update('test.tbc', {_id: 1}, {$set:
 {example:2}},
{'upsert':true, 'multi':false} );
if (x) {
   Network.metricAddCount('perf_trigger_success', 1);
}
else {
   Network.metricAddCount('perf_trigger_error', 1);
}
```

Référez-vous à http://docs.mongodb.org/manual/reference/method/db.collection.update/#db.collection.update pour plus d'informations.

Exemples de déclencheurs

- Exemple : analyse du syslog sur TCP avec une analyse de charge utile universelle

`Remote.Raw`

Le `Remote.Raw` la classe vous permet de soumettre des données brutes à un Raw flux de données ouvert (ODS) cible via un port TCP ou UDP.

Vous devez d'abord configurer une cible ODS brute à partir des paramètres d'administration, ce qui nécessite des privilèges d'administration du système et des accès. Pour plus d'informations sur la configuration, consultez Flux de données ouverts section du Guide de l'interface utilisateur ExtraHop Admin.

Note: Si la fonctionnalité Gzip est activée pour le flux de données brutes dans les paramètres d'administration, la classe Remote.Raw compresse automatiquement les données avec Gzip.

Méthodes

`send`

Envoie des données brutes à une cible Raw Open Data Stream (ODS) via un port TCP ou UDP.

Syntaxe :

```
Remote.Raw.send("data")
```

```
Remote.Raw("name").send("data")
```

Paramètres :

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`data:` ***Corde***

Chaîne JavaScript représentant les octets à envoyer.

Valeurs renvoyées :

Aucune

Exemples

```
Remote.Raw.send("data over the wire");
```

```
Remote.Raw("my-target").send("extra data for my-target");
```

`Remote.Syslog`

Le `Remote.Syslog` la classe vous permet de créer des messages Syslog distants et d'envoyer des données de message à un Syslog flux de données ouvert (ODS).

Vous devez d'abord configurer une cible Syslog ODS à partir des paramètres d'administration, ce qui nécessite des privilèges d'administration du système et des accès. Pour plus d'informations sur la configuration, consultez Flux de données ouverts section du Guide de l'interface utilisateur ExtraHop Admin.

Note: Si l'envoi d'un message rsyslog aboutit, les API renverront true. En cas de succès ou d'échec, le déclencheur continuera à s'exécuter car l'échec de l'envoi d'un message rsyslog est un échec « logiciel ». Une utilisation incorrecte des API, c'est-à-dire le fait de les appeler avec le mauvais nombre ou le mauvais type d'arguments, entraînera toujours l'arrêt de l'exécution du déclencheur.

Méthodes

`emerg(message:` ***Corde*** `):` ***vide***

Envoie un message au serveur Syslog distant avec un niveau de gravité d'urgence.

Syntaxe :

```
Remote.Syslog.emerg("eh_event=web uri=" + HTTP.uri + " req_size="
 + HTTP.reqSize + "
rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

```
Remote.Syslog("name").emerg("eh_event=web uri=" + HTTP.uri + "
 req_size=" +
```

```
HTTP.reqSize + " rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

Paramètres

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`alert(message:` ***Corde*** `):` ***vide***

Envoie un message au serveur Syslog distant avec un niveau de gravité de l'alerte.

Syntaxe :

```
Remote.Syslog.alert("eh_event=web uri=" + HTTP.uri + " req_size="
 + HTTP.reqSize + "
rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

```
Remote.Syslog("name").alert("eh_event=web uri=" + HTTP.uri + "
 req_size=" +
HTTP.reqSize + " rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

Paramètres

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`crit(message:` ***Corde*** `):` ***vide***

Envoie un message au serveur Syslog distant avec un niveau de gravité critique.

Syntaxe :

```
Remote.Syslog.crit("eh_event=web uri=" + HTTP.uri + " req_size=" +
 HTTP.reqSize + "
rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

```
Remote.Syslog("name").crit("eh_event=web uri=" + HTTP.uri + "
 req_size=" +
HTTP.reqSize + " rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

Paramètres

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`error(message:` ***Corde*** `):` ***vide***

Envoie un message au serveur Syslog distant avec un niveau de gravité de l'erreur.

Syntaxe :

```
Remote.Syslog.error("eh_event=web uri=" + HTTP.uri + " req_size="
 + HTTP.reqSize + "
rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

```
Remote.Syslog("name").error("eh_event=web uri=" + HTTP.uri + "
 req_size=" +
```

```
HTTP.reqSize + " rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

Paramètres

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`warn(message:` ***Corde*** `):` ***vide***

Envoie un message au serveur Syslog distant avec un niveau de gravité de l'avertissement.

Syntaxe :

```
Remote.Syslog.warn("eh_event=web uri=" + HTTP.uri + " req_size=" +
 HTTP.reqSize + "
rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

```
Remote.Syslog("name").warn("eh_event=web uri=" + HTTP.uri + "
 req_size=" +
HTTP.reqSize + " rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

Paramètres

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`notice(message:` ***Corde*** `):` ***vide***

Envoie un message au serveur Syslog distant avec un niveau de gravité de notification.

Syntaxe :

```
Remote.Syslog.notice("eh_event=web uri=" + HTTP.uri + " req_size="
 + HTTP.reqSize + "
rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

```
Remote.Syslog("name").notice("eh_event=web uri=" + HTTP.uri + "
 req_size=" +
HTTP.reqSize + " rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

Paramètres

`name:` ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`info(message:` ***Corde*** `):` ***vide***

Envoie un message au serveur Syslog distant avec un niveau de gravité des informations.

Syntaxe :

```
Remote.Syslog.info("eh_event=web uri=" + HTTP.uri + " req_size=" +
 HTTP.reqSize + "
rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

```
Remote.Syslog("name").info("eh_event=web uri=" + HTTP.uri + "
 req_size=" +
```

```
HTTP.reqSize + " rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

Paramètres

`name` : ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

`debug(message:` ***Corde*** `)` : ***vide***

Envoie un message au serveur Syslog distant avec un niveau de gravité de débogage.

Syntaxe :

```
Remote.Syslog.debug("eh_event=web uri=" + HTTP.uri + " req_size="
 + HTTP.reqSize + "
rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

```
Remote.Syslog("name").debug("eh_event=web uri=" + HTTP.uri + "
 req_size=" +
HTTP.reqSize + " rsp_size=" + HTTP.rspSize + " processingTime=" +
 HTTP.processingTime);
```

Paramètres

nom : ***Corde***

Nom de la cible ODS à laquelle les demandes sont envoyées. Si ce champ n'est pas spécifié, le nom est défini sur `default`.

Taille du message

Par défaut, le message envoyé au serveur distant est limité à 1 024 octets, y compris l'en-tête du message et la bande-annonce (si nécessaire). L'en-tête du message inclut toujours la priorité et l'horodateur, qui, ensemble, peuvent atteindre 30 octets.

Si vous disposez des privilèges d'administration du système et des accès, vous pouvez augmenter la taille des messages par défaut dans les paramètres d'administration. Cliquez **Configuration en cours d'exécution** dans la section Paramètres de l'appliance, puis cliquez sur **Modifier la configuration**. Accédez à la section « remote » et sous le nom de la cible ODS, tel que « rsyslog », ajoutez « message_length_max » comme indiqué dans l'exemple ci-dessous. Le paramètre « message_length_max » s'applique uniquement au message transmis aux API Remote.Syslog ; l'en-tête du message n'est pas pris en compte dans le maximum.

```
"remote": {
   "rsyslog": {
   "host": "hostname",
   "port": 54322,
   "ipproto": "tcp",
   "message_length_max": 4000
   }
}
```

Horodatage

Le format d'horodateur par défaut pour les messages rsyslog est UTC. Vous pouvez modifier l'horodateur en heure locale lorsque vous configurez le flux de données ouvert dans les paramètres d'administration.

Exemples de déclencheurs

- Exemple : envoyer les données de l'équipement découvert à un serveur Syslog distant
- Exemple : analyse du syslog sur TCP avec une analyse de charge utile universelle

- Exemple : correspondance des clés topnset

Remote

Le Remote cette classe vous permet d'envoyer des données à un syslog, à une base de données ou à un serveur tiers via un flux de données ouvert (ODS) et d'accéder aux réponses renvoyées par les cibles HTTP ODS.

Évènements

REMOTE_RESPONSE

S'exécute lorsque le système ExtraHop reçoit une réponse d'une cible HTTP ODS.

Note: Un déclencheur s'exécute sur l'événement REMOTE_RESPONSE uniquement s'il a créé la demande ODS à l'origine de la réponse.

Propriétés

response: ***Objet***

Objet contenant des informations issues de la Réponse HTTP renvoyée par la cible ODS. L'objet de réponse possède les propriétés suivantes :

statusCode: ***Numéro***

Le code dac.status renvoyé par la cible ODS.

body: ***Tampon***

Le corps de la Réponse HTTP envoyée par la cible ODS.

headers: ***Objet***

Objet contenant les en-têtes de la réponse HTTP envoyée par la cible ODS. Si la réponse contient plusieurs en-têtes portant le même nom, la valeur de l'en-tête est un tableau. Par exemple, si Set-Cookie est spécifié plusieurs fois dans la réponse, vous pouvez accéder au premier cookie en spécifiant Remote.response.headers["Set-Cookie"][0].

context: ***Objet*** | ***Corde*** | ***Numéro*** | ***Booléen*** | ***nul***

Les informations contextuelles spécifiées dans le fichier Remote.HTTP context paramètre lors de l' envoi de la demande ODS. Pour plus d'informations, voir Remote.HTTP .

Classes de banque de données

Les classes d'API Trigger présentées dans cette section vous permettent d'accéder aux métriques d'une banque de données, ou d'un pont.

Classe	Descriptif
AlertRecord	Vous permet d'accéder alerte informations sur `ALERT_RECORD_COMMIT` événements.
Dataset	Vous permet d'accéder au format RAW jeu de données valeurs et fournit une interface pour le calcul des percentiles.
MetricCycle	Vous permet de récupérer les métriques publiées pendant un intervalle de cycle métrique représenté par `METRIC_CYCLE_BEGIN`, `METRIC_CYCLE_END`, et `METRIC_RECORD_COMMIT` événements.
MetricRecord	Permet d'accéder à l'ensemble actuel de mesures sur `METRIC_RECORD_COMMIT` événements.
Sampleset	Vous permet de récupérer des données récapitulatives sur les métriques.
Topnset	Vous permet d'accéder aux données d'un ensemble de métriques regroupées par une clé, telle qu'un URI ou un client adresse IP.

`AlertRecord`

La classe AlertRecord vous permet d'accéder à alerte informations sur `ALERT_RECORD_COMMIT` événements.

Évènements

`ALERT_RECORD_COMMIT`

S'exécute lorsqu'une alerte se produit. Permet d'accéder aux informations relatives à l' alerte.

Des options de banque de données supplémentaires sont disponibles lorsque vous créez un déclencheur qui s'exécute sur cet événement. Voir Options de déclencheur avancées pour plus d' informations.

Note: Vous ne pouvez pas attribuer des déclencheurs qui s'exécutent uniquement lors de cet événement à des appareils ou à des groupes d'équipements spécifiques. Les déclencheurs qui s'exécutent lors de cet événement seront exécutés chaque fois que cet événement se produira.

Important: Cet événement s'exécute uniquement si le module NPM est activé sur le système ExtraHop. Si l'accès au module NPM n'a pas été autorisé à accéder à votre compte utilisateur, vous ne pouvez pas configurer de déclencheur pour qu'il s'exécute sur cet événement.

Propriétés

`description:` ***Corde***

Description de l'alerte telle qu'elle apparaît dans le système ExtraHop.

`id:` ***Corde***

L'ID de l'enregistrement d'alerte. Les ID d'enregistrement d'alerte sont nommés selon le format suivant :

```
extrahop.<object>.<alert_type>
```

`<object>` est le type d'objet auquel s'applique l'alerte. Pour les objets réseau, le `<object>` la valeur est capture. Si l'alerte concerne une métrique détaillée du topnset, le `<alert_type>` est `alert_detail`; sinon, le `<alert_type>` est `alert`. Les ID d'enregistrement d'alerte suivants sont valides :

- `extrahop.capture.alert`
- `extrahop.capture.alert_detail`
- `extrahop.device.alert`
- `extrahop.device.alert_detail`
- `extrahop.application.alert`
- `extrahop.application.alert_detail`
- `extrahop.flow_network.alert`
- `extrahop.flow_network.alert_detail`
- `extrahop.flow_interface.alert`
- `extrahop.flow_interface.alert_detail`

Note: Vous pouvez limiter l'exécution du déclencheur pour qu'il ne soit exécuté que pour des types d'enregistrement d'alerte spécifiques. Tapez une liste d'identifiants d'enregistrement d'alerte séparés par des virgules dans le **Types de métriques** champ des options avancées du déclencheur.

`name:` ***Corde***

Le nom de l'alerte.

`object:` ***Objet***

L'objet auquel s'applique l'alerte. Pour les alertes relatives à un équipement, à une application, à une capture, à une interface de flux ou à un réseau de flux, cette propriété contiendra un Device , Application , Network , FlowInterface , ou FlowNetwork objet, respectivement.

`time:` ***Numéro***

Heure à laquelle l'enregistrement d'alerte sera publié.

`severityName:` ***Corde***

Le nom du niveau de gravité de l'alerte. Les niveaux de gravité suivants sont pris en charge :

Valeur	Descriptif
`emerg`	Urgence
`alert`	Alerte
`crit`	Critique
`err`	Erreur
`warn`	Avertissement
`notice`	Remarque
`info`	Infos
`debug`	Déboguer

`severityLevel:` ***Numéro***

Niveau de gravité numérique de l'alerte. Les niveaux de gravité suivants sont pris en charge :

Valeur	Descriptif
0	Urgence
1	Alerte
2	Critique
3	Erreur
4	Avertissement
5	Remarque
6	Infos
7	Déboguer

Dataset

La classe Dataset vous permet d'accéder aux valeurs brutes des ensembles de données et fournit une interface pour calculer les percentiles.

Méthodes d'instance

`percentile(...):` ***Array*** | ***Numéro***

Accepte une liste de percentiles (sous forme de tableau ou d'arguments multiples) à calculer et renvoie les valeurs de percentiles calculées pour l'ensemble de données. Si un seul argument numérique est transmis, un nombre est renvoyé. Sinon, un tableau est renvoyé. Les arguments doivent être classés par ordre croissant, sans doublons. Les valeurs à virgule flottante, telles que 99,99, sont autorisées.

Propriétés de l'instance

`entries:` ***Array***

Tableau d'objets avec des attributs de fréquence et de valeur. Ceci est analogue à un tableau de fréquences où il existe un ensemble de valeurs et le nombre de fois que chaque valeur a été observée.

MetricCycle

Le `MetricCycle` class représente un intervalle pendant lequel les métriques sont publiées. La classe MetricCycle est valide le `METRIC_CYCLE_BEGIN`, `METRIC_CYCLE_END`, et `METRIC_RECORD_COMMIT` événements.

Le `METRIC_RECORD_COMMIT` l'événement est défini dans MetricRecord section.

Évènements

`METRIC_CYCLE_BEGIN`

S'exécute lorsqu'un intervalle métrique commence.

Note: Vous ne pouvez pas attribuer des déclencheurs qui s'exécutent uniquement lors de cet événement à des appareils ou à des groupes d'équipements spécifiques. Les déclencheurs qui s'exécutent lors de cet événement seront exécutés chaque fois que cet événement se produira.

`METRIC_CYCLE_END`

S'exécute à la fin d'un intervalle métrique.

Note: Vous ne pouvez pas attribuer des déclencheurs qui s'exécutent uniquement lors de cet événement à des appareils ou à des groupes d'équipements spécifiques. Les déclencheurs qui s'exécutent lors de cet événement seront exécutés chaque fois que cet événement se produira.

Des options de banque de données supplémentaires sont disponibles lorsque vous créez un déclencheur qui s'exécute sur l'un de ces événements. Voir Options de déclencheur avancées pour plus d' informations.

Propriétés

`id:` ***Corde***

Chaîne représentant le cycle métrique. La seule valeur possible est `30sec`.

`interval:` ***Objet***

Un objet contenant les propriétés de début et de fin, exprimées en millisecondes depuis l' époque.

`store:` ***Objet***

Un objet qui conserve des informations sur tous les `METRIC_RECORD_COMMIT` événements qui se produisent au cours d'un cycle métrique, c'est-à-dire à partir du `METRIC_CYCLE_BEGIN` événement au `METRIC_CYCLE_END` événement. Cet objet est analogue au `Flow.store` objet. Le `store` l'objet est partagé entre les déclencheurs pour `METRIC_* events`. Il est effacé à la fin d'un cycle métrique.

Exemples de déclencheurs

- Exemple : ajouter des métriques au magasin du cycle métrique

`MetricRecord`

Le `MetricRecord` la classe vous permet d'accéder à l'ensemble actuel de métriques sur `METRIC_RECORD_COMMIT` événements.

Évènements

`METRIC_RECORD_COMMIT`

S'exécute lorsqu'un enregistrement métrique est validé dans la banque de données et donne accès à diverses propriétés métriques.

Des options de banque de données supplémentaires sont disponibles lorsque vous créez un déclencheur qui s'exécute lors de cet événement. Voir Options de déclencheur avancées pour plus d'informations.

Note: Vous ne pouvez pas attribuer des déclencheurs qui s'exécutent uniquement lors de cet événement à des appareils ou à des groupes d'équipements spécifiques. Les déclencheurs qui s'exécutent lors de cet événement seront exécutés chaque fois que cet événement se produira.

Propriétés

`fields:` ***Objet***

Objet contenant des valeurs métriques. Les propriétés sont les noms des champs et les valeurs peuvent être des nombres, Topnset, Ensemble de données ou Set d'échantillons.

`id:` ***Corde***

Le type métrique, tel que `extrahop.device.http_server`.

`object:` ***Objet***

L'objet auquel la métrique s'applique. Pour les alertes d'équipement, d'application ou de VLAN, cette propriété contient un Device un objet, un Application un objet, ou un VLAN instance, respectivement. Pour les métriques de capture, telles que `extrahop.capture.net`, la propriété

contient un Network objet. L'exemple de code suivant stocke l'ID d'une application dans une variable :

```
var app_id = MetricRecord.object.id;
```

Note: L'exemple de code ci-dessus génère toujours l'avertissement suivant dans l'éditeur de déclencheur :

```
Property 'id' does not exist on type 'Device | Application
 | VLAN | Network'. ts(2339) [2, 33]
Property 'id' does not exist on type 'Network'.
```

L' avertissement indique que l'attribution du déclencheur à un réseau n'est pas prise en charge. Vous pouvez ignorer cet avertissement lorsque le déclencheur est attribué à une application.

`time:` ***Numéro***

Heure de publication de l'enregistrement métrique.

Exemples de déclencheurs

- Exemple : correspondance des clés topnset
- Exemple : ajouter des métriques au magasin du cycle métrique

`Sampleset`

La classe Sampleset vous permet de récupérer des données récapitulatives sur les métriques.

Propriétés

`count:` ***Numéro***

Le nombre d'échantillons dans le jeu d'échantillons.

`mean:` ***Numéro***

La valeur moyenne des échantillons.

`sigma:` ***Numéro***

L'écart type.

`sum:` ***Numéro***

Somme des échantillons.

`sum2:` ***Numéro***

Somme des carrés des échantillons.

`Topnset`

Le `Topnset` la classe représente un ensemble de métriques regroupées par une clé telle qu'un URI ou un client adresse IP.

Pour les métriques personnalisées, les touches du topnset correspond aux clés passées dans `metricAddDetail*()` méthodes. Les valeurs clés peuvent être un nombre, une chaîne, Dataset , Sampleset , ou un autre topnset.

`Methods`

`findEntries(key:` ***Adresse IP*** | ***Corde*** | ***Objet*** `):` ***Array***

Renvoie toutes les entrées dont les clés correspondent.

`findKeys(key:` ***Adresse IP*** | ***Corde*** | ***Objet*** `):` ***Array***

Renvoie toutes les clés correspondantes.

`lookup(key:` ***Adresse IP*** | ***Corde*** | ***Objet*** `):` ***

Recherchez un élément dans le topnset et récupérez la première entrée correspondante.

Propriétés

`entries:` ***Array***

Tableau des entrées du topnset. Le tableau contient au maximum `N` objets dotés de propriétés de clé et de valeur où `N` est actuellement fixé à 1000.

Clés dans le `entries` le tableau respecte la structure ou le modèle de clé suivant :

`type:` ***Corde***

Type de clé topnset. Les types de clés suivants sont pris en charge :

- `int`
- `string`
- `device_id`
- `ipaddr`
- `addr_pair`
- `ether`

`value:` ***

La valeur de la clé, qui varie en fonction du type de clé.

- Pour `int`, `string`, et `device_id` clés, la valeur est respectivement un nombre, une chaîne et un identifiant d'équipement.
- Pour `ipaddr` clés, la valeur est un objet contenant les propriétés suivantes :
  - `addr`
  - `proto`
  - `port`
  - `device_id`
  - `origin`
- Pour `addr_pair` clés, la valeur est un objet contenant les propriétés suivantes :
  - `addr1`
  - `addr2`
  - `port1`
  - `port2`
  - `proto`
- Pour `ether` clés, la valeur est un objet contenant les propriétés suivantes :
  - `ethertype`
  - `hwaddr`

Éléments d'API obsolètes

Les éléments d'API répertoriés dans cette section sont devenus obsolètes. Chaque élément inclut une alternative et la version dans laquelle l'élément est devenu obsolète.

Si votre script de déclencheur contient un élément obsolète, le validateur de syntaxe de l'éditeur de déclencheur vous indique quel élément est obsolète et suggère un élément de remplacement, s'il est disponible. Vous ne pouvez pas enregistrer le déclencheur tant que vous n'avez pas corrigé votre code ou désactivé la validation syntaxique. Pour améliorer les performances du déclencheur, remplacez les éléments obsolètes.

Options de déclencheur avancées obsolètes

Option	Remplacement	Version
`5min`, `1hr`, et `24hr` cycles métriques	Il n'est pas possible de remplacer les cycles métriques de 5 minutes, 1 heure et 24 heures. Cependant, les cycles métriques de 30 secondes sont toujours pris en charge.	9,6

Fonctions globales obsolètes

Fonction	Remplacement	Version
`exit(): Vide`	La déclaration de retour	4,0
`getTimestampMSec(): Numéro`	`getTimestamp(): Numéro`	4,0

Paramètres de fonction globaux obsolètes

Fonction	Propriété	Remplacement	Version
`commitDetection()`	`categories`	Tu peux spécifier les catégories de détection dans le catalogue de détection.	9,3

Événements obsolètes

Événement	Remplacement	Version
`NEW_VLAN`	Pas de remplacement	6.1

Classes obsolètes

Classe	Remplacement	Version
`RemoteSyslog`	Remote.Syslog	4,0
`XML`	Expressions régulières	6,0
`TroubleGroup`	Pas de remplacement	6,0

Méthodes obsolètes par classe

Classe	Méthode	Remplacement	Version
Flow	`getApplication(): Corde`	`getApplications(): Corde`	5.3
	`setApplication(name: Corde, turnTiming: Booléen): vide`	`addApplication(name: Corde, turnTiming: Booléen): vide`	5,3
Session	`update(key: Corde, value: *, options: Objet)*`	`replace(key: Corde, value: *, options: Objet): *`	3,9
SSL	`setApplication(name: Corde): vide`	`addApplication(name: Corde): vide`	5,3

Propriétés obsolètes par classe

Classe	Propriété	Remplacement	Version
AAA	`error: Corde`	`isError: Booléen`	5,0
	`tprocess: Numéro`	`processingTime: Numéro`	5,2
DB	`tprocess: Numéro`	`processingTime: Numéro`	5,2
Detection	`participants.object_type: Corde`	instance de l'opérateur	7,8
Discover	`vlan: VLAN`	Pas de remplacement	6.1
DNS	`tprocess: Numéro`	`processingTime: Numéro`	5,2
Flow	`isClientAborted: Booléen`	`isAborted: Booléen`	3,10
	`isServerAborted: Booléen`	`isAborted: Booléen`	3,10
	`turnInfo: Corde`	De haut niveau Turn objet avec des attributs pour le tour	3,9
FTP	`tprocess: Numéro`	`processingTime: Numéro`	5,2
HL7	processus : *Numéro*	Délai de traitement : *Numéro*	5,2
HTTP	`payloadText: Corde`	`payload: Tampon`	4,0
	`tprocess: Numéro`	`processingTime: Numéro`	5,2
IBMMQ	`messageID: Corde`	`msgID: Tampon`	5,2
	`msgSize: Numéro`	`totalMsgLength: Numéro`	5,2
	`objectHandle: Corde`	Pas de remplacement	5,0
	`payload: Tampon`	`msg: Tampon`	5,2
ICA	`authTicket: Corde`	`user: Corde`	3,7
	`application: Corde`	`program: Corde`	5,2
	`client: Corde`	`clientMachine: Corde`	6,0
LDAP	`tprocess: Numéro`	`processingTime: Numéro`	5,2
MongoDB	`tprocess: Numéro`	`processingTime: Numéro`	5,2
NetFlow	`tos: Numéro`	`dscp: Numéro`	6.1

Classe	Propriété	Remplacement	Version
		`dscp:` *Corde*	
NTLM	`ntlmRspVersion:` *Corde*	`rspVersion:` *Corde*	8,2
QUIC	`cyuFingerprint:` *Corde*	Pas de remplacement	9,6
	`tags:` *Tableau d'objets*	Pas de remplacement	9,6
	`record.cyuFingerprint:` *Corde*	Pas de remplacement	9,6
SMPP	`tprocess:` *Numéro*	`processingTime:` *Numéro*	5,2
SMTP	`recipient:` *Corde*	`recipientList:` *Tableau de cordes*	7,5
	`tprocess:` *Numéro*	`processingTime:` *Numéro*	5,2
SLL	`SSL.record.ja3Hash:` *Corde*	`SSL.ja3Hash:` *Corde*	9,7
	`SSL.record.ja3sHash:` *Corde*	`SSL.ja3sHash` *Corde*	9,7
	`reqBytes:` *Numéro*	`clientBytes:` *Numéro*	6.1
	`reqL2Bytes:` *Numéro*	`clientL2Bytes:` *Numéro*	6.1
	`reqPkts:` *Numéro*	`clientPkts:` *Numéro*	6.1
	`rspBytes:` *Numéro*	`serverBytes:` *Numéro*	6.1
	`rspL2Bytes:` *Numéro*	`serverL2Bytes:` *Numéro*	6.1
	`rspPkts:` *Numéro*	`serverPkts:` *Numéro*	6.1
TCP	`wndSize:` *Numéro*	`initRcvWndSize:` *Numéro*	6.2
	`wndSize1:` *Numéro*	`initRcvWndSize1:` *Numéro*	6.2
	`wndSize2:` *Numéro*	`initRcvWndSize2:` *Numéro*	6.2
Turn	`reqSize:` *Numéro*	`clientBytes:` *Numéro*	4,0
	`reqXfer:` *Numéro*	`clientTransferTime:` *Numéro*	4,0
	`respSize:` *Numéro*	`serverBytes:` *Numéro*	4,0
	`rspXfer:` *Numéro*	`serverTransferTime:` *Numéro*	4,0
	`tprocess:` *Numéro*	`processingTime:` *Numéro*	4,0

Options de déclencheur avancées

Vous pouvez configurer des options avancées pour certains événements lorsque vous créez un déclencheur.

Le tableau suivant décrit les options avancées disponibles et les événements applicables.

Option	Descriptif	Événements pris en charge
Octets par paquet à capturer	Spécifie le nombre d'octets à capturer par paquet. La capture commence par le premier octet du paquet. Spécifiez cette option uniquement si le script déclencheur effectue une capture de paquets. La valeur 0 indique que la capture doit collecter tous les octets de chaque paquet.	Tous les événements sont pris en charge à l'exception de la liste suivante : • `ALERT_RECORD_COMMIT` • `METRIC_CYCLE_BEGIN` • `METRIC_CYCLE_END` • `FLOW_REPORT` • `NEW_APPLICATION` • `NEW_DEVICE` • `SESSION_EXPIRE`
Octets de charge utile L7 vers la mémoire tampon	Spécifie le nombre maximum d'octets de charge utile à mettre en mémoire tampon. **Note:** Si plusieurs déclencheurs sont exécutés sur le même événement, le déclencheur dont la valeur L7 Payload Octets to Buffer est la plus élevée détermine la charge utile maximale pour cet événement pour chaque déclencheur.	• `CIFS_REQUEST` • `CIFS_RESPONSE` • `HTTP_REQUEST` • `HTTP_RESPONSE` • `ICA_TICK` • `LDAP_RESPONSE`
Octets du presse-papiers	Spécifie le nombre d'octets à mettre en mémoire tampon lors	• `ICA_TICK`

Option	Descriptif	Événements pris en charge
	d'un transfert dans le presse-papiers Citrix.	
Cycle métrique	Spécifie la durée du cycle métrique, exprimée en secondes. La seule valeur valide est `30sec`.	• `METRIC_CYCLE_BEGIN` • `METRIC_CYCLE_END` • `METRIC_RECORD_COMMIT`
Types de métriques	Spécifie le type de métrique par le nom brut de la métrique, tel que `extrahop.device.http_server`. Spécifiez plusieurs types de métriques dans une liste séparée par des virgules.	• `ALERT_RECORD_COMMIT` • `METRIC_RECORD_COMMIT`
Exécuter le déclencheur à chaque tour de flux	Permet la capture de paquets sur chaque flux tourner. L'analyse par tour analyse en continu la communication entre deux terminaux pour extraire un seul point de données de charge utile du flux. Si cette option est activée, toutes les valeurs spécifiées pour **Chaîne correspondant au client** et **Chaîne correspondante au serveur** les options sont ignorées.	• `SSL_PAYLOAD` • `TCP_PAYLOAD`
Plage de ports clients	Spécifie la plage de ports du client. Les valeurs valides sont comprises entre 0 et 65535.	• `SSL_PAYLOAD` • `TCP_PAYLOAD` • `UDP_PAYLOAD`
Octets du client vers la mémoire tampon	Spécifie le nombre d'octets du client à mettre en mémoire tampon. La valeur de cette option ne peut pas être définie sur 0 si la valeur du **Octets du serveur à mettre en mémoire tampon** l'option est également définie sur 0.	• `SSL_PAYLOAD` • `TCP_PAYLOAD`
Chaîne de recherche Client Buffer	Spécifie la chaîne de format qui indique quand commencer à mettre en mémoire tampon les données du client. Renvoie le paquet entier en cas de correspondance de chaîne. Vous pouvez spécifier la chaîne sous forme de texte ou de	• `SSL_PAYLOAD` • `TCP_PAYLOAD` • `UDP_PAYLOAD`

Option	Descriptif	Événements pris en charge
	nombres hexadécimaux. Par exemple, les deux `ExtraHop` et `\x45\x78\x74\x72\x61\x48\x6F\x70` sont équivalents. Les nombres hexadécimaux ne font pas la distinction entre majuscules et minuscules. Toute valeur spécifiée pour cette option est ignorée si **Par tour** ou **Exécuter le déclencheur sur tous les protocoles UDP** l'option de paquets est activée.	
Plage de ports du serveur	Spécifie la plage de ports du serveur. Les valeurs valides sont comprises entre 0 et 65535.	• SSL_PAYLOAD • TCP_PAYLOAD • UDP_PAYLOAD
Octets du serveur vers la mémoire tampon	Spécifie le nombre d'octets du serveur à mettre en mémoire tampon. La valeur de cette option ne peut pas être définie sur 0 si la valeur du **Octets du client à mettre en mémoire tampon** l'option est également définie sur 0.	• SSL_PAYLOAD • TCP_PAYLOAD
Chaîne de recherche dans la mémoire tampon du serveur	Spécifie la chaîne de format qui indique quand commencer à mettre en mémoire tampon les données du serveur. Vous pouvez spécifier la chaîne sous forme de texte ou de nombres hexadécimaux. Par exemple, les deux `ExtraHop` et `\x45\x78\x74\x72\x61\x48\x6F\x70` sont équivalents. Les nombres hexadécimaux ne font pas la distinction entre majuscules et minuscules. Toute valeur spécifiée pour cette option est ignorée si **Par tour** ou **Exécuter le déclencheur sur tous**	• SSL_PAYLOAD • TCP_PAYLOAD • UDP_PAYLOAD

Option	Descriptif	Événements pris en charge
	les protocoles UDP l'option est activée.	
Exécuter le déclencheur sur tous les paquets UDP	Permet la capture de tous les datagrammes UDP.	• `UDP_PAYLOAD`
Exécutez FLOW_CLASSIFY sur des flux non classés expirant	Permet de lancer l'événement à son expiration afin de cumuler des métriques pour flux qui n'étaient pas classés avant leur expiration.	• `FLOW_CLASSIFY`
Types externes	Spécifie les types de données externes que le déclencheur traite. Le déclencheur ne s'exécute que si la charge utile contient un champ de type avec l'une des valeurs spécifiées. Spécifiez plusieurs types dans une liste séparée par des virgules.	1. `EXTERNAL_DATA`

Exemples

Les exemples suivants sont disponibles :

Exemple : collecte de métriques ActiveMQ

Dans cet exemple, le déclencheur enregistre les informations de destination à partir du service de messagerie Java (JMS). Le déclencheur crée une application et collecte des métriques personnalisées indiquant notamment si le courtier d'un événement est l'expéditeur ou le destinataire et le champ de destination JMS spécifié pour cet événement.

Exécutez le déclencheur lors des événements suivants : `ACTIVEMQ_MESSAGE`

```
var app = Application("ActiveMQ Sample");
    if (ActiveMQ.senderIsBroker) {
       if (ActiveMQ.receiverIsBroker) {
          app.metricAddCount("amq_broker", 1);
          app.metricAddDetailCount("amq_broker", ActiveMQ.queue, 1);
       }
       else {
          app.metricAddCount("amq_msg_out", 1);
          app.metricAddDetailCount("amq_msg_out", ActiveMQ.queue, 1);
       }
}
else {
    app.metricAddCount("amq_msg_in", 1);
    app.metricAddDetailCount("amq_msg_in", ActiveMQ.queue, 1);
}
```

Cours connexes

- ActiveMQ
- Application

Exemple : envoyer des données à Azure avec Remote.http

Dans cet exemple, le déclencheur envoie des données au service de stockage Microsoft Azure Table via un flux de données ouvert (ODS) HTTP.

Vous devez d'abord configurer un flux de données ouvert HTTP à partir des paramètres d'administration avant de créer le déclencheur. La configuration ODS contient les informations d'authentification requises pour vous connecter à votre service Microsoft Azure. Pour plus d'informations sur la configuration, voir Configuration d'une cible HTTP pour un flux de données ouvert dans le Guide de l'interface utilisateur d'ExtraHop.

Exécutez le déclencheur lors des événements suivants : HTTP_RESPONSE

```
// The name of the HTTP destination defined in the ODS config
var REST_DEST = "my_table_storage";

// The name of the table within Azure Table storage
var TABLE_NAME = "TestTable";

/* If the header is not set to this value, Azure expects to receive XML;
 * however, it is easier for a trigger to send JSON.
 * The ODS config enables you to specify the datatype of fields; in this
 case
 * the timestamp (TS) field is a datetime even though it is serialized from
 a
 * Date to a String.
 */

var headers = { "Content-Type": "application/json;odata=minimalmetadata" };

var now = new Date(getTimestamp());
var msg = {
    "RowKey":        now.getTime().toString(), // must be a string
    "PartitionKey":  "my_key", // must be a string
    "HTTPMethod":    HTTP.method,
    "DestAddr":      Flow.server.ipaddr,
    "SrcAddr":       Flow.client.ipaddr,
    "SrcPort":       Flow.client.port,
    "DestPort":      Flow.server.port,
    "TS@odata.type": "Edm.DateTime", // metadata to describe format of TS
 field
    "TS":            now.toISOString(),
    "ServerTime":    HTTP.processingTime,
    "RspTTLB":       HTTP.rspTimeToLastByte,
    "RspCode":       HTTP.statusCode.toString(),
    "URI":           "http://" + HTTP.host + HTTP.path,
};

// debug(JSON.stringify(msg));
Remote.HTTP(REST_DEST).post( { path: "/" + TABLE_NAME, headers: headers,
 payload:
JSON.stringify(msg) } );
```

Cours connexes

- Remote.HTTP
- Flow
- HTTP

Exemple : Surveiller les actions des PME sur les appareils

Dans cet exemple, le déclencheur surveille les actions SMB effectuées sur les appareils, puis crée des mesures d'équipement personnalisées qui collectent le nombre total d'octets lus et écrits, ainsi que le nombre d'octets écrits par les utilisateurs SMB qui ne sont pas autorisés à accéder à une ressource sensible.

Exécutez le déclencheur sur les événements suivants : `CIFS_RESPONSE`

```
var client = Flow.client.device,
   server = Flow.server.device,
   clientAddress = Flow.client.ipaddr,
   serverAddress = Flow.server.ipaddr,
   file = CIFS.resource,
   user = CIFS.user,
   resource,
   permissions,
   writeBytes,
   readBytes;

// Resource to monitor
resource = "\\Clients\\Confidential\\";
// Users of interest and their permissions
permissions = {
   "\\\\EXTRAHOP\\tom" : {read: false, write: false},
   "\\\\Anonymous" : {read: true, write: false},
   "\\\\WORKGROUP\\maria" : {read: true, write: true}
};

// Check if this is an action on your monitored resource
if ((file !== null) && (file.indexOf(resource) !== -1)) {
   if (CIFS.isCommandWrite) {
      writeBytes = CIFS.reqSize;
      // Record bytes written
      Device.metricAddCount("cifs_write_bytes", writeBytes);
      Device.metricAddDetailCount("cifs_write_bytes", user, writeBytes);
      // Record number of writes
      Device.metricAddCount("cifs_writes", 1);
      Device.metricAddDetailCount("cifs_writes", user, 1);
      // Record number of unauthorized writes
      if (!permissions[user] || !permissions[user].write) {
         Device.metricAddCount("cifs_unauth_writes", 1);
         Device.metricAddDetailCount("cifs_unauth_writes", user, 1);
      }
   }

   if (CIFS.isCommandRead) {
      readBytes = CIFS.reqSize;
      // Record bytes read
      Device.metricAddCount("cifs_read_bytes", readBytes);
      Device.metricAddDetailCount("cifs_read_bytes", user, readBytes);
      // Record number of reads
      Device.metricAddCount("cifs_reads", 1);
      Device.metricAddDetailCount("cifs_reads", user, 1);
      // Record number of unauthorized reads
      if (!permissions[user] || !permissions[user].read) {
         Device.metricAddCount("cifs_unauth_reads", 1);
         Device.metricAddDetailCount("cifs_unauth_reads", user, 1);
      }
   }
}
```

Cours connexes

- CIFS
- Device
- Flow

Exemple : suivi des réponses HTTP de niveau 500 par ID client et URI

Dans cet exemple, le déclencheur suit les réponses du serveur HTTP qui génèrent un code d'erreur de 500. Le déclencheur crée également des métriques d'équipement personnalisées qui collectent l'identifiant du client et l'URI dans l'en-tête de chaque réponse 500.

Exécutez le déclencheur lors des événements suivants : HTTP_REQUEST et HTTP_RESPONSE

```
var custId,
   query,
   uri,
   key;

if (event === "HTTP_REQUEST") {
   custId = HTTP.headers["Cust-ID"];
   // Only keep the URI if there is a customer id
   if (custId !== null) {
      Flow.store.custId = custId;

      query = HTTP.query;

      /* Pull the complete URI (URI plus query string) and save it to
       * the Flow store for a subsequent response event.
       *
       * The query string data is only available on the request.
       */
      uri = HTTP.uri;
      if ((uri !== null) && (query !== null)) {
         uri = uri + "?" + query;
      }

      // Keep URIs for handling by HTTP_RESPONSE triggers
      Flow.store.uri = uri;
   }
}
else if (event === "HTTP_RESPONSE") {
   custId = Flow.store.custId;

   // Count total requests by customer ID
   Device.metricAddCount("custid_rsp_count", 1);
   Device.metricAddDetailCount("custid_rsp_count_detail", custId, 1);

   // If the status code is 500 or 503, record the URI and customer ID
   if ((HTTP.statusCode === 500) || (HTTP.statusCode === 503)){
      // Combine URI and customer ID to create the detail key
      key = custId;
      if (Flow.store.uri != null) {
        key += ", " + Flow.store.uri;
      }
      Device.metricAddCount("custid_error_count", 1);
      Device.metricAddDetailCount("custid_error_count_detail", key, 1);
   }
}
```

Cours connexes

- HTTP
- Flow
- Device

Exemple : collecte des mesures de réponse sur les requêtes de base de données

Dans cet exemple, le déclencheur crée des métriques d'équipement personnalisées qui collectent le nombre de réponses et les temps de traitement des requêtes de base de données.

Exécutez le déclencheur lors des événements suivants : `DB_RESPONSE`

```
let stmt = DB.statement;
if (stmt === null) {
    return;
}

// Remove leading whitespace and truncate
stmt = stmt.trimLeft().substr(0, 1023);

// Record counts by statement
Device.metricAddCount("db_rsp_count", 1);
Device.metricAddDetailCount("db_rsp_count_detail", stmt, 1);

// Record processing times by statement
Device.metricAddSampleset("db_proc_time", DB.processingTime);
Device.metricAddDetailSampleset("db_proc_time_detail",
                                stmt, DB.processingTime);
```

Cours connexes

- DB
- Device

Exemple : envoyer les données de l'équipement découvert à un serveur Syslog distant

Dans cet exemple, le déclencheur détecte lorsqu'un nouvel équipement est détecté sur le système ExtraHop et crée des messages Syslog distants contenant les attributs de l'équipement.

Vous devez d'abord configurer un flux de données ouvert à distance à partir des paramètres d'administration avant de créer le déclencheur. La configuration ODS indique l'emplacement du serveur Syslog distant. Pour plus d'informations sur la configuration, voir Configurer une cible Syslog pour un flux de données ouvert dans le Guide de l'interface utilisateur d'ExtraHop.

Exécutez le déclencheur lors des événements suivants : `NEW_DEVICE`

```
var dev = Discover.device;
Remote.Syslog.info('Discovered device ' + dev.id + ' (hwaddr: ' + dev.hwaddr
 + ')
');
```

Cours connexes

- Remote.Syslog
- Discover

- Device

Exemple : envoyer des données à Elasticsearch avec Remote.http

Dans cet exemple, le déclencheur envoie des données à un serveur Elasticsearch via un flux de données ouvert (ODS) HTTP.

Vous devez d'abord configurer un flux de données ouvert HTTP à partir des paramètres d'administration avant de créer le déclencheur. La configuration ODS spécifie la cible Elasticsearch et tous les identifiants d'authentification requis. Pour plus d'informations sur la configuration, voir Configuration d'une cible HTTP pour un flux de données ouvert dans le Guide de l'interface utilisateur d'ExtraHop.

Exécutez le déclencheur lors des événements suivants : HTTP_REQUEST et HTTP_RESPONSE

```
var date = new Date();
var payload = {
    'ts' : date.toISOString(), // Timestamp recognized by Elasticsearch
    'eh_event' : 'http',
    'my_path' : HTTP.path};
var obj = {
    'path' : '/extrahop/http', // Add to ExtraHop index
    'headers' : {},
    'payload' : JSON.stringify(payload)} ;
Remote.HTTP('elasticsearch').request('POST', obj);
```

Cours connexes

- Remote.HTTP

Exemple : accéder aux attributs d'en-tête HTTP

Dans cet exemple, le déclencheur accède aux attributs d'événement HTTP à partir de l'objet d'en-tête et crée des métriques d'équipement personnalisées qui comptent les demandes d'en-tête et les attributs.

Exécutez le déclencheur lors des événements suivants : HTTP_RESPONSE

```
var hdr,
   session,
   accept,
   results,
   headers = HTTP.headers,
   i;

// Header lookups are case-insensitive properties
session = headers["X-Session-Id"];

/* Session is a string representing the value of the header (or null
 * if the header is not present). Header values are always strings.
 */

// This syntax also works if the header is a legal property name
accept = headers.accept;

/*
 * In the event that there are multiple instances of a header,
 * accessing the header in the above manner (as a property)
 * will always return the value for the first appearance of the
 * header.
 */
```

```
if (session !== null)
{
   // Count requests per session ID
   Device.metricAddCount("req_count", 1);
   Device.metricAddDetailCount("req_count", session, 1);
}

/* Looping over all headers
 *
 * The "length" property is case-sensitive and is not
 * treated as a header lookup. Instead, it returns the number of
 * headers (as if HTTP.headers were an array). In the unlikely
 * event that there is a header called "Length," it would still be
 * accessible with HTTP.headers["Length"] (or HTTP.headers.Length).
 */

for (i = 0; i < headers.length; i++) {
   hdr = headers[i];
   debug("headers[" + i + "].name: " + hdr.name);
   debug("headers[" + i + "].value: " + hdr.value);
   Device.metricAddCount("hdr_count", 1);
   /* Count instances of each header */
   Device.metricAddDetailCount("hdr_count", hdr.name, 1);
}

// Searching for headers by prefix
results = HTTP.findHeaders("Content-");

/* The "results" property is an array (a real javascript array, as opposed
 * to an array-like object) of header objects (with name and value
 * properties) where the names match the prefix of the string passed
 * to findHeaders.
 */
for (i = 0; i < results.length; i++) {
   hdr = results[i];
   debug("results[" + i + "].name: " + hdr.name);
   debug("results[" + i + "].value: " + hdr.value);
}
```

Cours connexes

- HTTP
- Device

Exemple : collecte des métriques IBMMQ

Dans cet exemple, les déclencheurs fonctionnent ensemble pour donner une vue du flux de messages au niveau de la file d'attente via le IBMMQ protocole. Les déclencheurs créent des métriques d'application personnalisées qui comptent le nombre de messages entrants, sortants et échangés entre les courtiers par le biais de différentes files de messages.

Exécutez le déclencheur suivant sur `IBMMQ_REQUEST` événement.

```
if (IBMMQ.method == "MESSAGE_DATA") {
    var app = Application("IBMMQ Sample");
    app.metricAddCount("broker", 1);
    if (IBMMQ.queue !== null) {
        var ret = IBMMQ.queue.split(":");
        var queue = ret.length > 1 ? ret[1] : ret[0];
        app.metricAddDetailCount("broker", queue, 1);
    }
```

```
    else {
        app.metricAddCount("queueless_broker", 1);
    }
    if (IBMMQ.queue !== null && IBMMQ.queue.indexOf("QUEUE2") > -1) {
        app.metricAddCount("queue2_broker", 1);
    }
    app.commit();
}
elseif (IBMMQ.method == "MQPUT" || IBMMQ.method == "MQPUT1") {
    var app = Application("IBMMQ Sample");
    app.metricAddCount("msg_in", 1);
    if (IBMMQ.queue !== null) {
        var ret = IBMMQ.queue.split(":");
        var queue = ret.length > 1 ? ret[1] : ret[0];
        app.metricAddDetailCount("msg_in", queue, 1);
    }
    else {
        app.metricAddCount("queueless_msg_in", 1);
    }
    if (IBMMQ.queue !== null && IBMMQ.queue.indexOf("QUEUE2") > -1) {
        app.metricAddCount("queue2_msg_in", 1);
    }
    app.commit();
}
```

Exécutez le déclencheur suivant sur `IBMMQ_RESPONSE` événement.

```
if (IBMMQ.method == "ASYNC_MSG_V7" || IBMMQ.method == "MQGET_REPLY") {
    var app = Application("IBMMQ Sample");
    if (IBMMQ.payload === null) {
        app.metricAddCount("payloadless_msg_out", 1);
    }
    else {
        app.metricAddCount("msg_out", 1);
        if (IBMMQ.queue !== null) {
            var ret = IBMMQ.queue.split(":");
            var queue = ret.length > 1 ? ret[1] : ret[0];
            app.metricAddDetailCount("msg_out", queue, 1);
        }
        else {
            app.metricAddCount("queueless_msg_out", 1);
        }
        if (IBMMQ.queue !== null && IBMMQ.queue.indexOf("QUEUE2") > -1) {
            app.metricAddCount("queue2_msg_out", 1);
        }
    }
    app.commit();
}
```

Cours connexes

- IBMMQ
- Application

Exemple : enregistrer les succès et les échecs de Memcache

Dans cet exemple, le déclencheur crée des métriques d'équipement personnalisées qui enregistrent chaque cache mémoire hit or miss et le temps d'accès de chaque accès.

Exécutez le déclencheur lors des événements suivants : MEMCACHE_RESPONSE

```
var hits = Memcache.hits;
var misses = Memcache.misses;
var accessTime = Memcache.accessTime;
var i;

Device.metricAddCount('memcache_key_hit', hits.length);

for (i = 0; i < hits.length; i++) {
   var hit = hits[i];
   if (hit.key != null) {
      Device.metricAddDetailCount('memcache_key_hit_detail', hit.key, 1);
   }
}

if (!isNaN(accessTime)) {
   Device.metricAddSampleset('memcache_key_hit', accessTime);
   if ((hits.length > 0) && (hits[0].key != null)) {
      Device.metricAddDetailSampleset('memcache_key_hit_detail',
 hits[0].key,
                                      accessTime);
   }
}

Device.metricAddCount('memcache_key_miss', misses.length);

for (i = 0; i < misses.length; i++) {
   var miss = misses[i];
   if (miss.key != null) {
      Device.metricAddDetailCount('memcache_key_miss_detail', miss.key, 1);
   }
}
```

Cours connexes

- Memcache
- Device

Exemple : analyse des clés de cache mémoire

Analyse le cache mémoire clés pour extraire des ventilations détaillées, par exemple par module d'identification et nom de classe, et créer des métriques d'équipement personnalisées pour collecter des informations clés.

Les clés sont formatées comme suit : `"com.extrahop.<module>.<class>_<id>"` —par exemple : `"com.extrahop.widgets.sprocket_12345"`.

Exécutez le déclencheur lors des événements suivants : MEMCACHE_RESPONSE

```
var method = Memcache.method;
var statusCode = Memcache.statusCode;
var reqKeys = Memcache.reqKeys;
var hits = Memcache.hits;
var misses = Memcache.misses;
var error = Memcache.error;
var hit;
var miss;
var key;
var size;
var reqKey;
```

```
var i;

// Record breakdown of hit count and value size by module and class
for (i = 0; i < hits.length; i++) {
   hit = hits[i];
   key = hit.key;
   size = hit.size;

   Device.metricAddCount("hit", 1);
   if (key != null) {
      var parts = key.split(".");

      if ((parts.length == 4) && (parts[0] == "com") &&
         (parts[1] == "extrahop")) {
         var module = parts[2];
         var subparts = parts[3].split("_");

         Device.metricAddDetailCount("hit_module", module, 1);
         Device.metricAddDetailSampleset("hit_module_size", module, size);

         if (subparts.length == 2) {
            var hitClass = module + "." + subparts[0];

            Device.metricAddDetailCount("hit_class", hitClass, 1);
            Device.metricAddDetailSampleset("hit_class_size", hitClass,
                                            size);
         }
      }
   }
}

// Record misses by ID to help identify caching issues
for (i = 0; i < misses.length; i++) {
   miss = misses[i];
   key = miss.key;
   if (key != null) {
      var parts = key.split(".");

      if ((parts.length == 4) && (parts[0] == "com") &&
         (parts[1] == "extrahop") && (parts[2] == "widgets")) {
         var subparts = parts[3].split("_");

         if ((subparts.length == 2) && (subparts[0] == "sprocket")) {
            Device.metricAddDetailCount("sprocket_miss_id", subparts[1], 1);
         }
      }
   }
}

// Record the keys that produced any errors
if (error != null && method != null) {
   for (i = 0; i < reqKeys.length; i++) {
      reqKey = reqKeys[i];
      if (reqKey != null) {
         var errDetail = method + " " + reqKey + " / " + statusCode + ": " +
                         error;
         Device.metricAddDetailCount("error_key", errDetail, 1);
      }
   }
}

// Record the status code, matching built-in metrics
if (Memcache.isBinaryProtocol && statusCode != "NO_ERROR") {
   Device.metricAddDetailCount("status_code",
```

```
                                    method + "/" + statusCode, 1);
}
else {
   Device.metricAddDetailCount("status_code", statusCode, 1);
}
```

Cours connexes

- Memcache
- Device

Exemple : ajouter des métriques au magasin du cycle métrique

Le déclencheur de cet exemple montre comment stocker temporairement les données de tous les validations d'enregistrements métriques effectuées au cours d'un cycle métrique.

Exécutez le déclencheur lors des événements suivants : `METRIC_CYCLE_BEGIN`, `METRIC_CYCLE_END`, `METRIC_RECORD_COMMIT`

Configurez options de déclencheur avancées comme indiqué dans le tableau suivant :

Option	Valeur
Cycle métrique	30 secondes
Type métrique	extrahop.device.http_server, extrahop.device.tcp

```
var store = MetricCycle.store;

function processMetric() {
    var id = MetricRecord.id,
        deviceId = MetricRecord.object.id,
        fields = MetricRecord.fields;

    if (!store.metrics[deviceId]) {
        store.metrics[deviceId] = {};
    }
    if (id === 'extrahop.device.http_server') {
        store.metrics[deviceId].httpRspAborted= fields['rsp_abort'];
    }
    else if (id === 'extrahop.device.tcp') {
        store.metrics[deviceId].tcpAborted = fields['aborted_out'];
    }
}

function commitSyntheticMetrics() {
    var dev,
        metrics,
        abortPct,
        deviceId;
    for (deviceId in store.metrics) {
        metrics = store.metrics[deviceId];
        abortPct = (metrics.httpRspAborted / metrics.tcpAborted) * 100;
        dev = new Device(deviceId);
        dev.metricAddSnap('http-tcp-abort-pct', abortPct);
    }
}

switch (event) {
```

```
case 'METRIC_CYCLE_BEGIN':
    store.metrics = {};
    break;

case 'METRIC_RECORD_COMMIT':
    processMetric();
    break;

case 'METRIC_CYCLE_END':
    commitSyntheticMetrics();
    break;
}
```

Cours connexes

- MetricCycle
- MetricRecord
- Device

Exemple : analyse de messages PoS personnalisés avec une analyse de charge utile universelle

Dans cet exemple, le déclencheur analyse les messages TCP provenant d'un système de point de vente (PoS) et crée des métriques d'équipement personnalisées qui collectent des valeurs spécifiques dans les 4e à 7e octets des messages de réponse et de demande.

Exécutez le déclencheur lors des événements suivants : TCP_PAYLOAD

```
// Define variables; store client or server payload into a Buffer object

var buf_client = Flow.client.payload,
    buf_server = Flow.server.payload,
    protocol = Flow.l7proto,

// PoS Message Type Structure Definition
    pos_message_type = {
        "0100" : "0100_Authorization_Request",
        "0101" : "0101_Authorization_Request_Repeat",
        "0110" : "0110_Authorization_Response",
        "0200" : "0200_Financial_Request",
        "0201" : "0201_Financial_Request_Repeat",
        "0210" : "0210_Financial_Response",
        "0220" : "0220_Financial_Transaction_Advice_Request",
        "0221" : "0221_Financial_Transaction_Advice_Request_Repeat",
        "0230" : "0230_Financial_Transaction_Advice_Response",
        "0420" : "0420_Reversal_Advice_Request",
        "0421" : "0421_Reversal_Advice_Request_Repeat",
        "0430" : "0430_Reversal_Advice_Response",
        "0600" : "0600_Administration_Request",
        "0601" : "0601_Administration_Request_Repeat",
        "0610" : "0610_Administration_Response",
        "0620" : "0620_Administration_Advice_Request",
        "0621" : "0621_Administration_Advice_Request_Repeat",
        "0630" : "0630_Administration_Advice_Response",
        "0800" : "0800_Administration_Request",
        "0801" : "0801_Administration_Request_Repeat",
        "0810" : "0810_Administration_Response"
    };

// Skip parsing if it is a protocol of no interest or there is no payload
```

```
if (protocol !== 'tcp:4015' || (buf_client === null && buf_server === null))
 {
    // debug('Protocol of no interest: ' + protocol);
    return;
} else {
    /* Store the data into variables for future access since there is some
 payload
     * to parse
     */
    var client_ip = Flow.client.ipaddr,
        server_ip = Flow.server.ipaddr,
        client_port = Flow.client.port,
        server_port = Flow.server.port;
        // client = new Device(Flow.client.device.id),
        // server = new Device(Flow.server.device.id);
}

if (buf_client !== null && buf_client.length >= 7) {

    // This is a client payload
    var cli_msg_type = buf_client.slice(3,7).decode('utf-8');
    debug('Client: ' + client_ip + ":" + client_port + " Type: " +
 pos_message_type[cli_msg_type]);
    Device.metricAddCount('UPA_Request', 1);
    Device.metricAddDetailCount('UPA_Request_by_Message',
 pos_message_type[cli_msg_type], 1);
    Device.metricAddDetailCount('UPA_Request_by_Client',
 client_ip.toString(), 1);

} else if (buf_server !== null && buf_server.length >= 7) {

    // This is a server payload
    var srv_msg_type = buf_server.slice(3,7).decode('utf-8');
    debug('Server: ' + server_ip + " Client: " + client_ip + ":" +
 client_port +"
Type: " + pos_message_type[srv_msg_type]);
    Device.metricAddCount('UPA_Response', 1);
    Device.metricAddDetailCount('UPA_Response_by_Message',
 pos_message_type[srv_msg_type], 1);
    Device.metricAddDetailCount('UPA_Response_by_Client',
 client_ip.toString(), 1);

} else {

    // No buffer captured situation
    //debug('Null or not enough buffer data');
    return;
}
```

Cours connexes

- Tampon
- Device
- Flow

Exemple : analyse du syslog sur TCP avec une analyse de charge utile universelle

Dans cet exemple, le déclencheur analyse le syslog via TCP et compte l' activité du syslog au fil du temps, à la fois à l'échelle du réseau et par équipement.

Note: Vous devrez peut-être modifier l'exemple du déclencheur pour vous assurer que les ports réseau de votre serveur Syslog correspondent aux ports de votre environnement.

Cet exemple de déclencheur est disponible en téléchargement via un bundle de solutions sur le Communauté ExtraHop.

Exécutez le déclencheur lors des événements suivants : TCP_PAYLOAD, UDP_PAYLOAD

```
// Global variables
var buffer         = Flow.client.payload,
    buffer_size    = Flow.client.payload.length + 1,
    client         = new Device(Flow.client.device.id),
    data_as_json   = { client_ip       : Flow.client.ipaddr.toString(),
                       client_port     : Flow.client.port.toString(),
                       server_ip       : Flow.server.ipaddr.toString(),
                       server_port     : Flow.server.port.toString(),
                       protocol        : 'syslog',
                       protocol_fields : {} },
    protocol        = Flow.l7proto,
    server          = new Device(Flow.server.device.id),
    syslog          = {},
    syslog_facility = {
        "0": "kern",
        "1": "user",
        "2": "mail",
        "3": "daemon",
        "4": "auth",
        "5": "syslog",
        "6": "lpr",
        "7": "news",
        "8": "uucp",
        "9": "clock_daemon",
        "10": "authpriv",
        "11": "ftp",
        "12": "ntp",
        "13": "log_audit",
        "14": "log_alert",
        "15": "cron",
        "16": "local0",
        "17": "local1",
        "18": "local2",
        "19": "local3",
        "20": "local4",
        "21": "local5",
        "22": "local6",
        "23": "local7",
    },
    syslog_priority = {
         "0": "emerg",
         "1": "alert",
         "2": "crit",
         "3": "err",
         "4": "warn",
         "5": "notice",
         "6": "info",
         "7": "debug",
    };

// Exit out early if not classified properly or no payload

if ( ( protocol != 'tcp:5141' ) || ( buffer === null ) ) {
    debug('Invalid protocol ' + protocol +
          ' or null buffer (' + buffer.unpack('z').join(' ') + ')');
    return;
```

```
}

// Get started parsing Syslog

var data = buffer.unpack('z');

// Separate the PRIO field from the rest of the message
var msg_part  = data[0].split('>')[1].split(' ');
var prio_part = data[0].split('>')[0].split('<')[1];

// Decode the PRIO field into Syslog facility and priority
var raw_facility = parseInt(prio_part) >> 3;
var raw_priority = parseInt(prio_part) & 7;

syslog.facility = syslog_facility[raw_facility];
syslog.priority = syslog_priority[raw_priority];

/* Timestamp and hostname are technically part of the HEADER field, but
 * treat the rest of the message as a <space> delimited
 * string, which it is (the syslog protocol is very basic)
 */
syslog.timestamp = msg_part.slice(0,3).join(' ');
syslog.hostname  = msg_part[3];
syslog.message   = msg_part.slice(4).join(' ');

/* At the network level, keep counts of who is sending messages by
 * both facility and priority
 */
Network.metricAddCount('syslog:priority_' + syslog.priority, 1);
Network.metricAddDetailCount('syslog:priority_' +
                             syslog.priority + '_detail',
                             Flow.client.ipaddr, 1);
Network.metricAddCount('syslog:facility_' + syslog.facility, 1);
Network.metricAddDetailCount('syslog:facility_' +
                             syslog.facility + '_detail',
                             Flow.client.ipaddr, 1);

/* Devices receiving messages keep a count of who sent those messages
 * by facility and priority
 */
server.metricAddCount('syslog:priority_' + syslog.priority, 1);
server.metricAddDetailCount('syslog:priority_' +
                            syslog.priority + '_detail',
                            Flow.client.ipaddr, 1);
server.metricAddCount('syslog:facility_' + syslog.facility, 1);
server.metricAddDetailCount('syslog:facility_' +
                            syslog.facility + '_detail',
                            Flow.client.ipaddr, 1);

/* Devices sending messages keep a count of who they sent those messages
 * to by facility and priority
 */
client.metricAddCount('syslog:priority_' + syslog.priority, 1);
client.metricAddDetailCount('syslog:priority_' +
                            syslog.priority + '_detail',
                            Flow.server.ipaddr, 1);
client.metricAddCount('syslog:facility_' + syslog.facility, 1);
client.metricAddDetailCount('syslog:facility_' +
                            syslog.facility + '_detail',
                            Flow.server.ipaddr, 1);

data_as_json.protocol_fields = syslog;
data_as_json.ts              = new Date();
```

```
//try {
//    Remote.MongoDB.insert('payload.syslog', data_as_json);
//}
//catch ( err ) {
//    Remote.Syslog.debug(JSON.stringify(data_as_json));
//}
debug('Syslog data: ' + JSON.stringify(data_as_json, null, 4));
```

Cours connexes

- Flow
- Network
- Tampon
- Remote.MongoDB
- Remote.Syslog

Exemple : analyse NTP avec analyse de charge utile universelle

Dans l'exemple suivant, le déclencheur analyse le protocole horaire du réseau par le biais d'une analyse de charge utile universelle (UPA).

Exécutez le déclencheur lors des événements suivants : UDP_PAYLOAD

```
var buf = Flow.server.payload,
    flags,
    values,
    fmt,
    offset = 0,
    ntpData = {},
    proto = Flow.l7proto;
if ((proto !== 'NTP') || (buf === null)) {
    return;
}
// Parse individual flag values from flags byte
function parseFlags(flags) {
    return {
        'LI': flags >> 6,
        'VN': (flags & 0x3f) >> 3,
        'mode': flags & 0x7
    };
}

// Convert from NTP short format
function ntpShort(n) {
    return n / 65536.0;
}

// Convert integral part of NTP timestamp format to Date
function ntpTimestamp(n) {
    /* NTP dates start at 1900, subtract the difference
     * and convert to milliseconds */
    var ms = (n - 0x83aa7e80) * 1000;
    return new Date(ms);
}

// First part of NTP header
fmt = ('B' + // Flags (LI, VN, mode)
       'B' + // Stratum
       'b' + // Polling interval (signed)
       'b' + // Precision (signed)
```

```
        'I' + // Root delay
        'I'); // Root dispersion

values = buf.unpack(fmt);

offset = values.bytes;

flags = parseFlags(values[0]);
if (flags.VN !== 4) {
    // Expecting NTPv4
    return;
}

ntpData.flags = flags;
ntpData.stratum = values[1];
ntpData.poll = values[2];
ntpData.precision = values[3];
ntpData.rootDelay = ntpShort(values[4]);
ntpData.rootDispersion = ntpShort(values[5]);

// The next field, the reference ID, depends upon the stratum field
switch (ntpData.stratum)
{
case 0:
case 1:
    // Identifier string (4 bytes), and 4 NTP timestamps in two parts
    fmt = '4s8I';
    break;
default:
    // Unsigned int (based on IP), and 4 NTP timestamps in two parts
    fmt = 'I8I';
    break;
}
// Passing in offset enables you to continue parsing where you left off
values = buf.unpack(fmt, offset);
ntpData.referenceId = values[0];

// Only the integral parts of the timestamp are referenced here
ntpData.referenceTimestamp = ntpTimestamp(values[1]);
ntpData.originTimestamp = ntpTimestamp(values[3]);
ntpData.receiveTimestamp = ntpTimestamp(values[5]);
ntpData.transmitTimestamp = ntpTimestamp(values[7]);

debug('NTP data:' + JSON.stringify(ntpData, null, 4));
```

Cours connexes

- Tampon
- Flow
- UDP

Exemple : enregistrer des données dans une table de session

Dans cet exemple, le déclencheur enregistre des transactions HTTP spécifiques dans la table de session et crée des métriques réseau personnalisées qui collectent les données d'expiration de session.

Exécutez le déclencheur lors des événements suivants : HTTP_REQUEST, SESSION_EXPIRE

```
// HTTP_REQUEST
if (event == "HTTP_REQUEST") {
   if (HTTP.userAgent === null) {
```

```
      return;
   }

   // Look for the OS name
   var re = /(Windows|Mac|Linux)/;
   var os = HTTP.userAgent.match(re);
   if (os === null) {
      return;
   }
   // Specify the matched string as the key for session table entry
   var os_name = os[0];

   var opts =
    {
      // Expire added entries after 30 seconds
      expire: 30,
      // Retain entries with normal priority if session table grows too
 large
      priority: Session.PRIORITY_NORMAL,
      // Make expired entries available on SESSION_EXPIRE events
      notify: true
   };
   // Ensure an entry for this key is present; an existing entry will not be
 replaced
   Session.add(os_name, 0, opts);
   // Increase the count for this entry
   var count = Session.increment(os_name);
   debug(os_name + ": " + count);
}

/* After 30 seconds, the accumulated per-OS counts appear in the
 Session.expiredKeys
 * list, accessible in the SESSION_EXPIRE event:
 */
   //SESSION_EXPIRE
if (event == "SESSION_EXPIRE"){
   var keys = Session.expiredKeys;
   for (var i = 0; i < keys.length; i++) {
      debug("count of " + keys[i].name + ": " + keys[i].value);
      if (keys[i].value > 500) {
         Network.metricAddCount("os-high-request-count", 1);
         Network.metricAddDetailCount("os-high-request-count",
                                      keys[i].name, 1);
      }
   }
}
```

Cours connexes

- HTTP
- Network
- Session

Exemple : suivre les requêtes SOAP

Dans cet exemple, le déclencheur suit les requêtes SOAP via l'en-tête SOAPAction, les enregistre dans le magasin de flux et crée des métriques réseau personnalisées qui collectent des données sur les transactions.

Note: Avant de commencer, vérifiez que votre implémentation SOAP transmet les informations nécessaires par le biais de l'en-tête.

Exécutez le déclencheur lors des événements suivants : HTTP_REQUEST, HTTP_RESPONSE

```
var soapAction,
   headers = HTTP.headers,
   method,
   detailMethod,
   parts;

if (event === "HTTP_REQUEST") {
   soapAction = headers["SOAPAction"]
   if (soapAction != null) {
       Flow.store.soapAction = soapAction;
   }
}
else if (event === "HTTP_RESPONSE") {
   soapAction = Flow.store.soapAction;
   if (soapAction != null) {
      parts = soapAction.split("/");
      if (parts.length > 0) {
         method = soapAction.split("/")[1];
      }
      else {
         method = soapAction;
      }
      detailMethod = method + "_detail";
      Network.metricAddCount(method, 1);
      Network.metricAddDetailCount(detailMethod, Flow.client.ipaddr, 1);
      Network.metricAddSampleset("soap_proc", HTTP.processingTime);
      Network.metricAddDetailSampleset("soap_proc_detail", method,
                                       HTTP.processingTime);
   }
}
```

Cours connexes

- Flow
- HTTP
- Network

Exemple : correspondance des clés topnset

Les déclencheurs de cet exemple illustrent la correspondance des touches topnset par chaîne et adresse IP , et incluent un mappage de touches avancé.

Correspondance des clés Topnset par chaîne

Exécutez le déclencheur lors des événements suivants : METRIC_RECORD_COMMIT

Configurez options de déclencheur avancées comme indiqué dans le tableau suivant :

Option	Valeur
Cycle métrique	30 secondes
Type métrique	extrahop.device.app

```
var stat = MetricRecord.fields['bytes_out'],
    id = MetricRecord.object.id,
    proto = 'HTTP2-SSL',
```

```
    entry;

entry = stat.lookup(proto);
if (entry !==null) {
    debug('Device ' + id + ' sent ' + entry.value + ' bytes over ' + proto);
}
```

Correspondance des clés Topnset par adresse IP

Exécutez le déclencheur lors des événements suivants : `METRIC_RECORD_COMMIT`

Configurez options de déclencheur avancées comme indiqué dans le tableau suivant :

Option	Valeur
Cycle métrique	30 secondes
Type métrique	extrahop.device.net_detail

```
var stat = MetricRecord.fields['bytes_out'],
    total = 0,
    entry,
    entries,
    i,
    ip = new IPAddress('192.168.112.1');

entries = stat.findEntries(ip);
for (i = 0; i < entries.length; i++) {
    entry = entries[i];
    total += entry.value;
}
Remote.Syslog.alert('IP ' + ip + ' sent ' + total + ' bytes.');
```

Correspondance avancée entre les touches du topnset

Exécutez le déclencheur lors des événements suivants : `METRIC_RECORD_COMMIT`

Configurez options de déclencheur avancées comme indiqué dans le tableau suivant :

Option	Valeur
Cycle métrique	30 secondes
Type métrique	extrahop.device.net_detail

```
var stat = MetricRecord.fields['bytes_out'],
    entry,
    entries,
    key,
    i;

entries = stat.findEntries({addr: /192.168.112.1*/, proto: 17});

debug('matched ' + entries.length + '/' + stat.entries.length + '
 entries')};

for (i = 0; i < entries.length; i++) {
    entry = entries[i];
    key = entry.key;
    Remote.Syslog.alert('unexpected outbound UDP traffic from: ' +
                        JSON.stringify(key));
```

```
}
```

Cours connexes

- MetricRecord
- IPAddress
- Remote.Syslog

Exemple : création d'un conteneur d'applications

Dans cet exemple, le déclencheur crée un conteneur d'applications basé sur le trafic associé à une application à deux niveaux et crée des métriques d'application personnalisées collectées sur le protocole HTTP et des événements de base de données.

Exécutez le déclencheur lors des événements suivants : HTTP_RESPONSE et DB_RESPONSE

```
/* Initialize the application object against which you will
 * commit specific HTTP and DB transactions. After traffic is
 * committed, an application container called "My App" will appear
 * in the Applications tab in the ExtraHop system.
 */

var myApp = Application("My App");

/* These configurable properties describe features that define
 * your application traffic.
 */

var myAppHTTPHost = "myapp.internal.example.com";
var myAppDatabaseName = "myappdb";
if (event == "HTTP_RESPONSE") {

   /* HTTP transactions can be committed to the application on
    * HTTP_RESPONSE events.
    */

   /* Commit this HTTP transaction only if the HTTP host header for
    * this response is defined and matches your application's HTTP host.
    */

   if (HTTP.host && (HTTP.host == myAppHTTPHost)) {
      myApp.commit();

      /* Capture custom metrics about user agents that experience
       * HTTP 40x or 50x responses.
       */

      if (HTTP.statusCode && (HTTP.statusCode >= 400))
{

         // Increment the overall count of 40x or 50x responses

         myApp.metricAddCount('myapp_40x_50x', 1);

         // Collect additional detail on referer, if any

         if (HTTP.referer) {
            myApp.metricAddDetailCount('myapp_40x_50x_refer_detail',
                                       HTTP.referer, 1);
         }
      }
```

```
    }

} else if (event == "DB_RESPONSE") {
    /* Database transactions can be committed to the application on
     * DB_RESPONSE events.
     *
     * Commit this database transaction only if the database name for
     * this response matches the name of our application database.
     */
    if (DB.database && (DB.database == myAppDatabaseName)) {
        myApp.commit();
    }
}
```

Cours connexes

- Application
- DB
- HTTP