cmap_keys

Vue générale dans le CMAP

Description

La librairie cmap est utilisée pour interagir avec la base de configuration utilisée par corosync. Il y a 3 types de clé stockés dans cmap : Mappage de valeurs stockées dans le fichier de configuration, statistiques temps réels, valeurs créées par l'utilisateur.

Clés

internal_configuration.* Données de configuration interne. Ces clés sont lecture seule. Si une configuration spécifique au sous-système est nécessaire, la clé doit être sous la forme **logging.logger_subsys.SERVICE.key** où SERVICE est un nom de service en majuscule et la clé est la même que dans le fichier de configuration.

nodelist.* Valeurs lues depuis le fichier de configuration. Chaque élément node dans le fichier de configuration à une position assignées commençant à 0. Donc de premier nœud a le préfixe **nodelist.node.0.**.

runtime.blackbox.* Trigger keys. Il est recommandé d'utiliser corosync-blackbox pour trouver facilement les adresses nodeid/ring du nœud local directement de cmap.

runtime.connections.* Informations sur le nombre total de connexions actives, terminées, et sur chaque connexion IPC à un moment donné. Tous les clés sont lecture seules.

runtime.connections.ID.* Chaque connexion IPC a un ID unique. C'est sous la forme **[[short_name :][PID :]internal_id**. Les clés typiques sont :

client_pid Contient le PID d'une connexion IPC

dispatched avec le nombre de messages dispatchés

invalid_request nombre de requête faites par IPC qui sont invalides

name Contenant le nom court d'une connexion IPC

overload Nombre de de requêtes qui n'ont pas été traité à cause d'une surcharge

queue_size nombre de messages en queue d'attente d'envoie

recv_retries nombre total de réceptions interrompues

requests nombre de requêtes faites par IPC

responses nombre de réponses envoyées par IPC

send_retries nombre total d'envoies interrompues

service_id ID du service auquel IPC est connecté

runtime.services.* statistiques pour les moteurs de service. Chaque service a son propre service_id avec un nom runtime.services.SERVICE., où SERVICE est un nom de service en minuscule.

runtime.totem.pg.mrp.srp.* Statistiques sur totem. Les clés sont lecture seule :

commit_entered Nombre de fois qu'un processeur entre en état COMMIT

commit_token_lost Nombre de fois qu'un processeur perd le token en état COMMIT

consensus_timeouts Nombre de fois qu'un processeur dépasse le délai en créant un consensus sur le membership

continuous_gather Nombre de fois qu'un processeur n'a pas été capable d'atteindre de consensus

firewall_enabled_or_nic_failure à 1 quand le processeur n'a pas été capable d'atteindre le consensus depuis longtemps.

gather_entered Nombre de fois qu'un processeur entre à l'état GATHER

gather_token_lost Nombre de fois qu'un processeur perd le token en état GATHER

mcast_retx Nombre de messages retransmis

mcast_rx Nombre de message multicast reçus

mcast_tx Nombre de message multicast transmis

memb_commit_token_rx Nombre de token reçus

memb_commit_token_tx Nombre de token émis

memb_join_rx Nombre de message join reçus

memb_join_tx Nombre de message join transmis

memb_merge_detect_rx Nombre de message member merge reçus

memb_merge_detect_tx Nombre de message member merge transmis

orf_token_rx Nombre de tokens orf reçus

orf_token_tx Nombre de tokens orf transmis

recovery_entered Nombre de fois qu'un processeur entre à l'état recovery

recovery_token_lost Nombre de fois qu'un token a été perdu à l'état recovery

rx_msg_dropped Nombre de messages reçus qui ont été supprimés parce qu'ils n'étaient pas attendus

token_hold_cancel_rx Nombre de token reçus contenant un message cancel

token_hold_cancel_tx Nombre de token émis contenant un message cancel

mtt_rx_token Mean Transmis Time du token, en ms. temps entre 2 tokens consécutifs reçus

avg_token_workload Temps moyen en ms du temps à maintenir le token dans le processeur courant

avg_backlog_calc Temps moyen de message non encore envoyés du processeur courant.

runtime.totem.pg.mrp.srp.members.* Contient les membres du protocole totem. Chaque membre à le format runtime.totem.pg.mrp.srp.members.NODEID.KEY œu key peut être :

ip L'adresse IP du membre

joint_count Nombre de fois que le processeur à joins le membership avec le processeur local.

status Statut du processeur

config_version Version de configuration du nœud membre.

resources.process.PID.* Préfixe créé par les applications utilisant SAM avec l'intégration CMAP. Il contient les clés suivantes :

recovery Stratégie de récupération du processus.

poll_period Valeur passée dans sam_initialize comme time_interval

last_updated Dernière fois que sam a reçus un heartbeat du client

state État du client.

uidgid.* Les informations sur les utilisateurs/groupes qui sont autorisés à faire des connexions IPC à corosync

quorum.cancel_wait_for_all Dit à votequorum d'annuler l'attente de tous les nœuds au démarrage du cluster. Peut être utilisé pour débloquer le quorum si des nœud sont éteinds.

config.reload_in_progress À 1 quand corosync.conf est rechargé, 0 sinon.

config.totemconfig_reload_in_progress Idem, mais changé après une configuration totem.

Exemples

Changer dynamiquement les permission user/group pour utiliser cororsync ipc :
corosync-cmapctl -s uidgid.uid.500 u8 1
GID est similaire :
corosync-cmapctl -s uidgid.gid.500 u8 1
Pour supprimer la permission, la supprimer :
corosync-cmapctl -d uidgid.gid.500

Ajouter/supprimer dynamiquement un nœud UDPU. On ajoute un nœud avec l'ip 10.34.38.108 et nodeid 3. On cherche une position de nœud qui n'est pas utilisée :

nodelist.local_node_pos (u32) = 1
nodelist.node.0.nodeid (u32) = 1
nodelist.node.0.ring0_addr (str) = 10.34.38.106
nodelist.node.1.nodeid (u32) = 2
nodelist.node.1.ring0_addr (str) = 10.34.38.107

Donc la position de nœud suivant sera 2 :

corosync-cmapctl -s nodelist.node.2.nodeid u32 3
corosync-cmapctl -s nodelist.node.2.ring0_addr str 10.34.38.108

Toujours ajouter ring0_addr en dernier