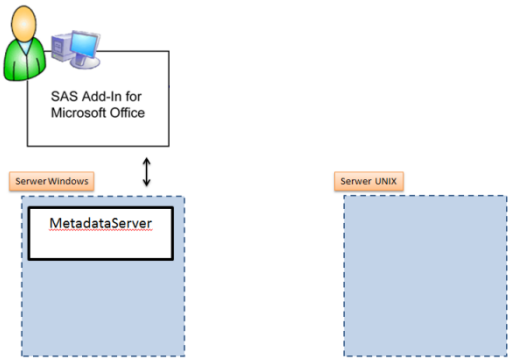
WorkspaceServer aspekty administracyjne

1 Wstęp

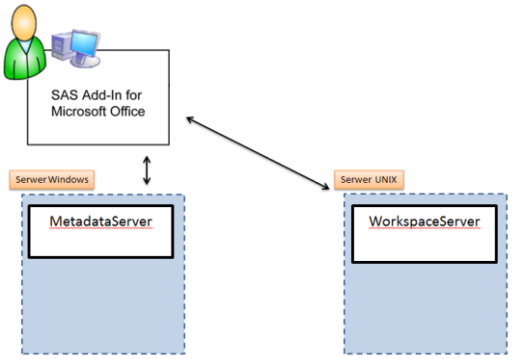
Niniejszy dokument przeznaczony jest dla administratorów środowisk SAS®9 oraz administratorów systemowych (Windows, Unix). Opisano tu sposób identyfikacji użytkownika końcowego odpowiedzialnego za określoną sesję SAS działającą na serwerze i potencjalnie obciążającą jego zasoby.

W środowiskach SAS®9, w których serwer metadanych umieszczony jest na innej maszynie niż serwer obliczeniowy, często na tej drugiej zachodzi potrzeba stworzenia kont użytkowników, w imieniu których powoływane są procesy SAS.

Użytkownicy aplikacji klienckich SAS (np. SAS Enterprise Guide, SAS Add-in for Microsoft Office, SAS Enterprise Miner) po uruchomieniu swojej aplikacji łączą się najpierw i logują do serwera metadanych. Wstępne logowanie polega na uwierzytelnieniu w systemie uwierzytelniania, z którego korzysta serwer metadanych, czyli najczęściej w systemie operacyjnym (np. Windows) bądź w Active Directory (ewentualnie w LDAP). W metadanych aplikacje klienckie znajdują informacje m.in. o dostępnych serwerach roboczych. Na potrzeby dalszych rozważań, przyjmujemy, iż serwerem obliczeniowym jest maszyna UNIX, co jest dość często spotykaną konfiguracją (nic nie stoi na przeszkodzie, aby był to osobny serwer Windows).



Na tym etapie na serwerze obliczeniowym jeszcze nic się nie dzieje. Dopiero próba otworzenia danych bądź uruchomienia wybranej analizy powoduje, iż nawiązywane jest połączenie do maszyny z serwerami roboczymi i startowany jest tzw. WorkspaceServer (WS). WS to nic innego jak proces SAS w systemie operacyjnym, uruchomiony na konkretnego użytkownika. Dla każdego użytkownika aplikacji klienckiej tworzony jest osobny proces WS.

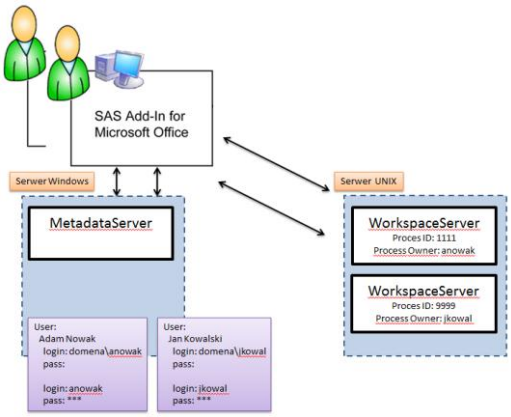


W imieniu jakiego użytkownika tworzony jest proces na maszynie obliczeniowej? Zawsze będzie to użytkownik widoczny dla UNIX-a, gdyż to system operacyjny musi być w stanie powołać w jego imieniu proces. W jaki sposób wybierany jest ten użytkownik? W zależności od konfiguracji środowiska metadanych możliwe są następujące scenariusze:

a. Aplikacja kliencka SAS zapyta się o poświadczenia (login oraz hasło użytkownika systemowego na UNIX); uwaga – nie pomylić z poświadczeniami dla serwera, na którym działają metadane.
b. Poświadczenia zostaną pobrane automatycznie z:
    i. serwera metadanych – jako atrybuty użytkownika (login indywidualny)
    ii. serwera metadanych – jako atrybuty grupy, której członkiem jest dany użytkownik (tzw. login grupowy)
    iii. serwera metadanych – jako atrybuty serwera (tzw. launch credential) przy WS skonfigurowanym do tzw. autentykacji token-owej.
    iv. tzw. „credentials cache" aplikacji – gdyż zostały uprzednio podane przez użytkownika lub pobrane z metadanych w trakcie trwania tej samej sesji aplikacji

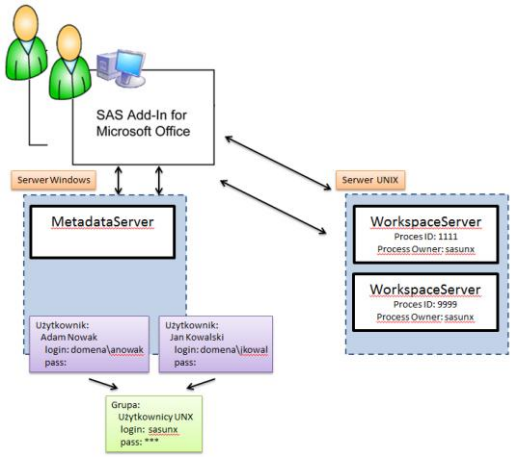
W tym momencie rodzi się kolejne pytanie: czy każdy użytkownik korzystający np. z Enterprise Guide powinien mieć założone konto systemowe na UNIX? Może, ale nie musi!

Może, ale wiąże się to potencjalnie z zakładaniem dużej liczby kont UNIX-owych, co nie zawsze przypada do gustu administratorom tego systemu. Zalety tego rozwiązania to: identyfikowalność użytkowników i ich procesów w systemie oraz możliwość precyzyjnego nadawania uprawnień w systemie plików. Taka konfiguracja odpowiada scenariuszom b.i lub b.iv.

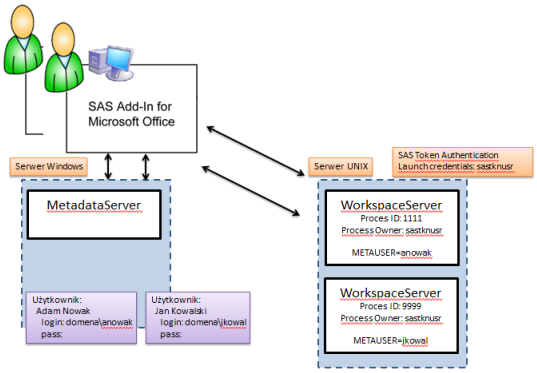


Drugim wyjściem jest konfiguracja PAM na serwerze UNIX, co oznacza możliwość logowania się do systemu UNIX z wykorzystaniem kont Active Directory (lub LDAP) – z reguły tych samych, z których korzysta serwer metadanych na Windows. Tym sposobem unikamy duplikowania systemowych baz użytkowników. Zalety są identyczne jak przy indywidualnych kontach UNIX-owych. Wadą: konieczność dodatkowej konfiguracji PAM.

Trzecim sposobem jest stworzenie pojedynczego konta UNIX-owego i umożliwienie, aby z tego jednego konta korzystali wszyscy użytkownicy aplikacji SAS. Jakkolwiek brzmi to dość „niebezpiecznie", jest to często stosowane rozwiązanie (scenariusz b.ii lub b.iii powyżej), gdy poświadczenia pobierane są automatycznie, a użytkownicy nawet nie są świadomi nawiązanego połączenia na serwer UNIX. Zaletą jest prostota konfiguracji w środowisku SAS oraz minimalna liczba kont w systemie operacyjnym. Wady to: brak możliwości granulacji uprawnień do zasobów systemu plików, gdyż procesy SAS poszczególnych użytkowników „metadanowych" wywoływane są na wspólne konto UNIX-owe. Z tym wiąże się utrudniona identyfikacja tychże procesów SAS i zmapowania ich na faktycznych użytkowników aplikacyjnych.



Rys. 2.1.1.a. WS z loginem grupowym



Rys. 2.1.1.b. WS w autentykacji tokenowej

2 Jak przyporządkować sesję WS użytkownikowi aplikacji SAS?

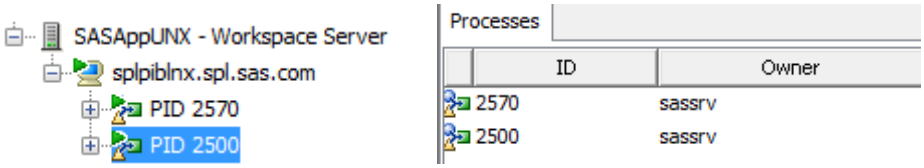
Pytanie to jest istotne w przypadku wyżej wymienionych scenariuszy b.ii (login grupowy) oraz b.iii (autentykacja tokenowa). Zacznijmy może od tego drugiego scenariusza czyli WS skonfigurowanego w autentykacji tokenowej, gdyż identyfikacja użytkowników aplikacyjnych wydaje się prostsza.

2.1.1 WS w autentykacji tokenowej

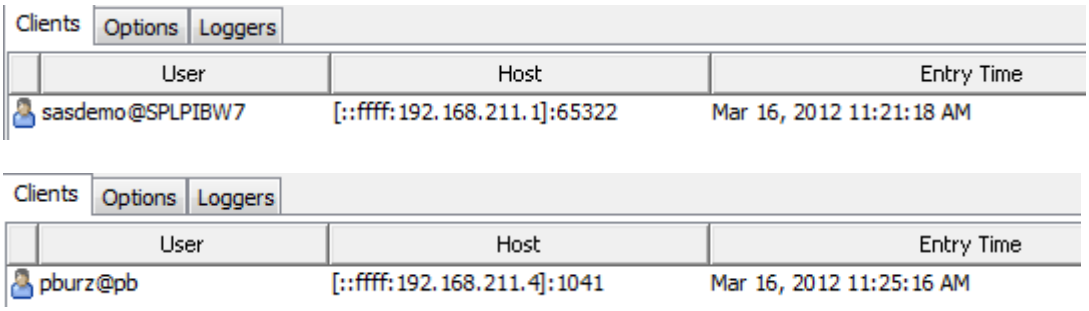
Jest to konfiguracja, którą można stosować od wersji SAS 9.2. Na pozór wydaje się identyczna z konfiguracją b.ii. Istotnie, z punktu widzenia systemu operacyjnego procesy użytkownika aplikacyjnego są powoływane na jedno konto systemowe. Jednakże różnica polega m.in. na tym, że sesja WS „tokenowego" jest świadoma dla jakiego użytkownika aplikacyjnego została powołana (ObjectSpawner przekazuje do powoływanej przez siebie sesji WS zmienną środowiskową METAUSER; sesja WS ma ustawioną makrozmienną _METAUSER, którą może wykorzystywać w kodzie 4GL). Sesja „tokenowego" WS będzie miała dostęp do bibliotek (i innych zasobów metadanowych), do których uprawnienia ma użytkownik aplikacyjny. W szczególności, na starcie

zostaną podpiętej tylko te biblioteki typu pre-assigned, do których uprawnienia ma użytkownik aplikacyjny. WS z loginem grupowym (konfiguracja b.ii) zachowuje się inaczej, o czym poniżej.

Administrator środowiska może przy pomocy SAS Management Console (SASMC) wyświetlić aktywne sesje WS: PID procesów, jego właściciela oraz - po wybraniu konkretnego procesu – użytkownika aplikacyjnego, adres IP komputera na którym działa aplikacja kliencka i czas powołania sesji. Tym sposobem jednoznacznie przyporządkowuje określony proces WS w systemie operacyjnym z aktywnym użytkownikiem końcowym. Obrazują to poniższe rysunki.



Rys. 2.1.1.a. Widok aktywnych sesji WS: dwa procesy działające w systemie operacyjnym w imieniu użytkownika 'sassrv', gdyż taki login został zdefiniowany jako tzw. launch credential dla WS „tokenowego"



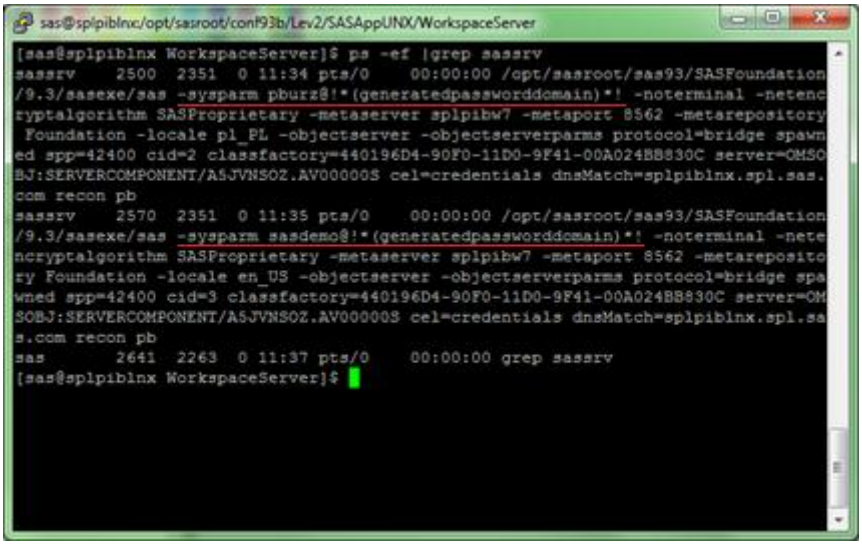
Rys. 2.1.1.b. Szczegółowy podgląd dla każdej z powyższych dwu sesji WS: informacja o użytkowniku aplikacyjnym, adresie IP spod którego nastąpiło połączenie i czas rozpoczęcia sesji.

W praktyce administratorzy często listują aktywne procesy, które obciążają maszynę (np. poleceniem 'ps –ef'). Listing procesów wyświetli wprawdzie PID, właściciela (sassrv), komendę wywołania SAS – ale w celu zmapowania procesów na użytkownika aplikacyjnego należy zapamiętać PIDy i sprawdzić przyporządkowanie w SASMC. Wydaje się to karkołomne. Czy istnieje sposób na szybsze zidentyfikowanie użytkownika aplikacyjnego? Jeżeli przypomnimy sobie, iż ObjectSpawner przekazuje do WS zmienną środowiskową METAUSER, to możemy wykorzystać ten fakt i dokonać drobnej modyfikacji skryptu …/Lev1/SASApp/WorkspaceServer/WorkspaceServer_usermods.sh, wpisując poniższy wiersz:

```
USERMODS_OPTIONS="-sysparm \"$METAUSER\""
```

Tym sposobem do wiersza wywołania sesji WS doklejona zostanie opcja „-sysparm" z zawartością zmiennej środowiskowej METAUSER, czyli id użytkownika aplikacyjnego. Listując procesy systemowe zobaczymy dodatkową opcję z właściwym użytkownikiem. (Opcja SAS '-sysparm <parametr>' służy do przekazywania poprzez wiersz poleceń dowolnego <parametru> do sesji SAS; przekazany parametr dostępny jest pod makrozmienną &SYSPARM. Jednak w naszym przypadku nie zależy nam na

wykorzystaniu tej makrozmiennej wewnątrz sesji WS, a jedynie na doklejeniu opcji do wywołania SAS).



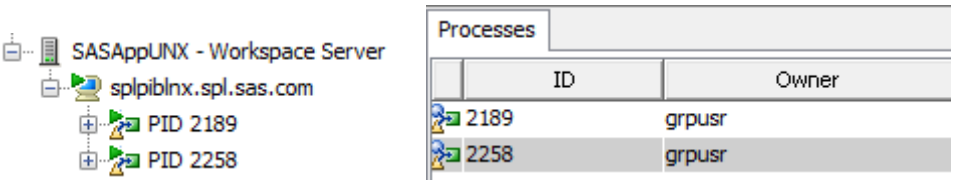
Rys. 2.1.1.c. Systemowy listing procesów „tokenowego" WS działających w imieniu 'sassrv' z opcją „sysparm" przekazującą użytkownika aplikacyjnego

Takie bezpośrednie przyporządkowanie procesu WS konkretnemu użytkownikowi aplikacyjnemu stwarza dodatkowe możliwości administracyjne, np. regularne gromadzenie informacji o wykorzystaniu zasobów (cpu, ram) przez poszczególnych użytkowników końcowych.

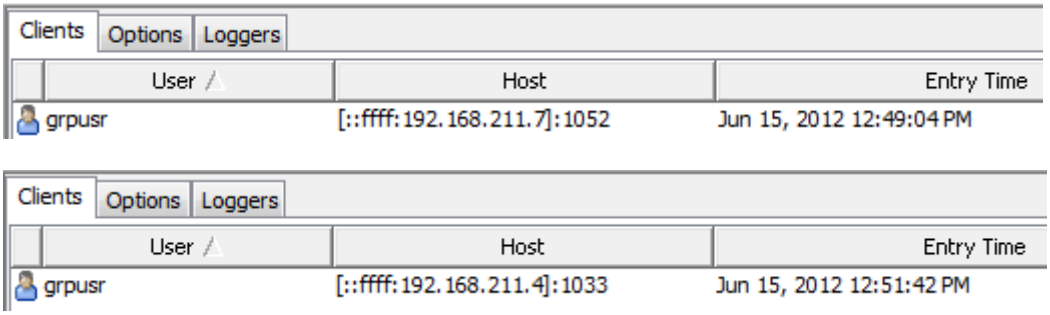
2.1.2 WS z loginem grupowym

Powróćmy w naszych rozważaniach do konfiguracji, która była często stosowana w środowiskach SAS 9.1.3, gdy nie istniała opcja autentykacji tokenowej. Konfiguracja polega na utworzeniu nowej grupy użytkowników w metadanych (np. Unix Users), przypisaniu tej grupie loginu z hasłem (np. grpusr) w domenie autoryzacji, z którą skojarzony jest WS. Kolejny krok to zapewnienie członkostwa w grupie wszystkim tym użytkownikom aplikacyjnym, którzy mają uruchamiać zadania na serwerze UNIX. Członkostwo w grupie gwarantuje bowiem widoczność loginu potrzebnego do uruchomienia sesji WS.

Uruchamiana w tym trybie sesja WS tworzy w systemie operacyjnym proces na użytkownika grupowego 'grpusr' czyli identycznie jak dla WS „tokenowego". Na czym więc polega różnica? Istotna różnica polega na tym, iż sesja „grupowego" WS nie jest świadoma dla jakiego użytkownika aplikacyjnego została powołana – a precyzyjniej mówiąc, sesja ta działa w imieniu grupy aplikacyjnej 'Unix Users'. ObjectSpawner, który powołuje sesje WS, przekazuje w zmiennej środowiskowej $METAUSER login skojarzony z grupą, a nie indywidualny login użytkownika aplikacyjnego. W związku z tym już na etapie powoływania procesu WS gubiona jest identyfikacja indywidualnego użytkownika. Ma to swoje praktyczne konsekwencje, np.: grupowy WS podepnie na starcie tylko te biblioteki typu pre-assigned, do których dostęp ma grupa 'Unix Users'. WS nie jest w stanie określić dla jakiego użytkownika aplikacyjnego został powołany. On został powołany dla grupy aplikacyjnej i w jej imieniu działa.

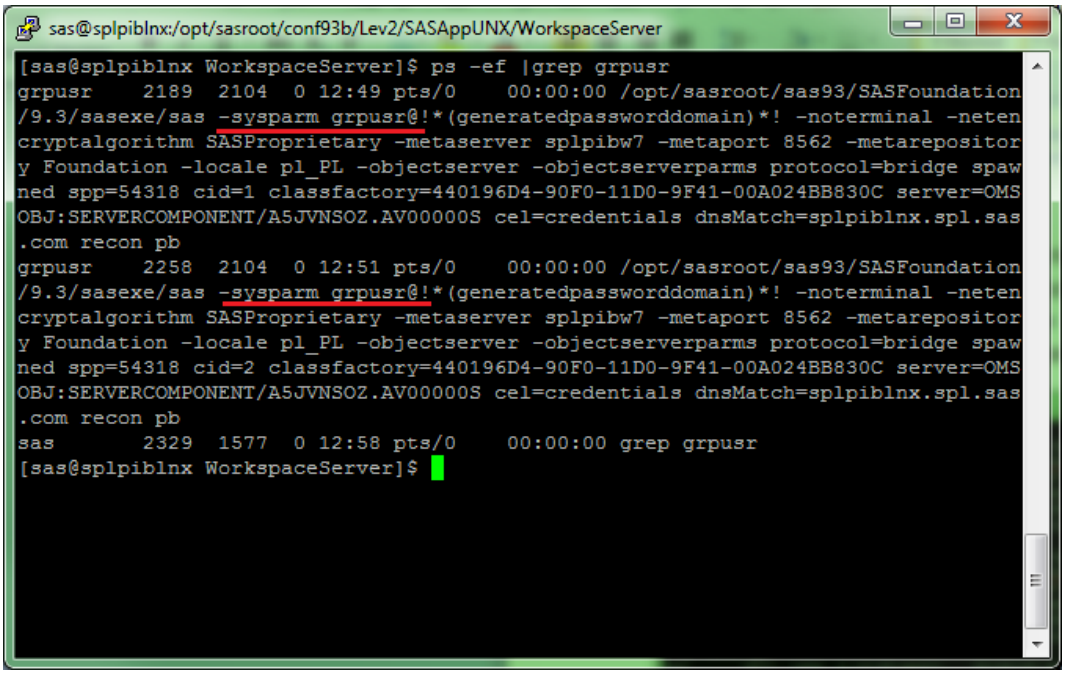


Rys. 2.1.2.a. Widok aktywnych sesji WS: dwa procesy działające w systemie operacyjnym w imieniu użytkownika 'grpusr', gdyż taki login został zdefiniowany grupie 'Unix Users' w domenie autoryzacji UnixAuth skojarzonej z WS.



Rys. 2.1.2.b. Szczegółowy podgląd dla każdej z powyższych dwu sesji WS: informacja o użytkowniku grupowym, adresie IP spod którego nastąpiło połączenie i czas rozpoczęcia sesji.

Zmienna systemowa $METAUSER zawiera teraz login grupowy, więc na jej podstawie nie można rozróżnić procesów dla poszczególnych użytkowników aplikacyjnych. W związku z tym traci sens wspomniana powyżej modyfikacja skryptu
…/Lev1/SASApp/WorkspaceServer/WorkspaceServer_usermods.sh



Rys. 2.1.2.c. Systemowy listing procesów „grupowego" WS

Czy istnieje zatem sposób na przyporządkowanie poszczególnym sesjom WS użytkowników aplikacyjnych? Tak.

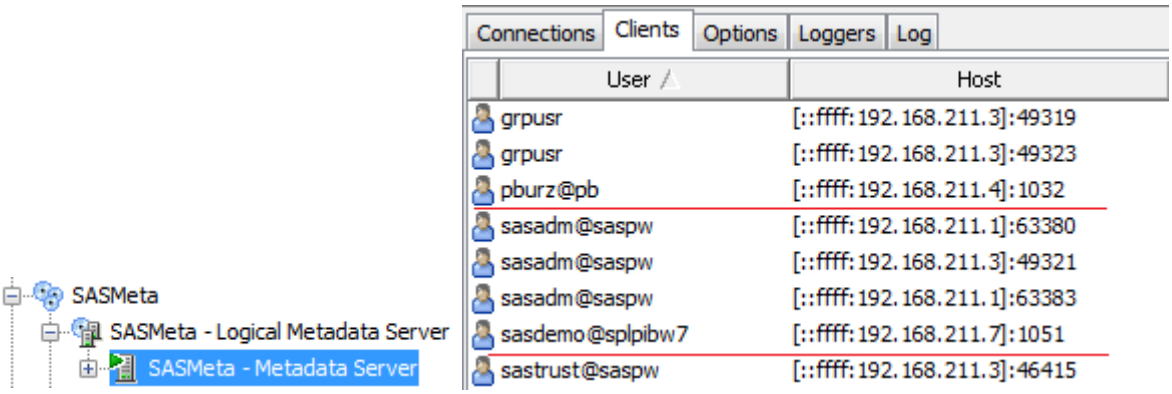
Można to określić pośrednio na podstawie adresu IP stacji klienckiej, widocznego albo z poziomu SASMC (przykład na Rys. 2.1.2.b) albo w logu WS. Ta druga opcja wymaga jednak włączenia logowania dla WS, które w domyślnej konfiguracji jest wyłączone. W szczególności należy ustawić logger Audit.Authentication (w pliku logconfig.xml dla WS) na poziomie "INFO". W logu WS zobaczymy wówczas wiersze jak na przykładzie poniżej.

```
2012-06-15T12:51:42,897 INFO  [00000074] :grpusr - Inherited client
connection (2) for user grpusr.  Encryption level is Credentials using
encryption algorithm SASProprietary.  Peer IP address and port are
[::ffff:192.168.211.4]:1033.
```

Jednakże aby nie korzystać z systemowych mapowań adres IP-użytkownik, można posłużyć się dodatkowo logiem serwera metadanych, w którym znajdziemy poniższe wiersze z momentu uruchamiania aplikacji klienckiej. Bazując na wspólnym kluczu jakim jest adres IP, odczytamy z poniższego logu login użytkownika aplikacyjnego.

```
2012-06-18T09:36:24,818 INFO [00003235] :pburz@pb - New client
connection (19) accepted from server port 8562 for user pburz@pb.
Encryption level is Credentials using encryption algorithm
SASPROPRIETARY. Peer IP address and port are
[::ffff:192.168.211.4]:1032.
```

Te same informacje można na bieżąco odczytać również z SASMC po połączeniu do serwera metadanych. Odnosząc się do adresu IP połączenia, możemy sprawdzić jaki użytkownik aplikacyjny jest z nim skojarzony.



Rys. 2.1.2.d. Lista aktywnych klientów połączonych do serwera metadanych

Przyglądając się powyższemu rysunkowi uważny czytelnik może zadać pytanie o dwa pierwsze połączenia na liście, wykonane w imieniu użytkownika 'grpusr' – skąd one się biorą? Są to połączenia wykonane bezpośrednio przez sesję WS na serwerze UNIX (stąd wspólny adres 192.168.211.3), która

podczas inicjalizacji łączy się z serwerem metadanych w celu odczytania bibliotek typu ‘preassigned’. Tymczasem zaznaczone podkreśleniami połączenia spod adresów 192.168.211.4 oraz 192.168.211.7 zostały nawiązane przez aplikacje klienckie. I dzięki nim jesteśmy w stanie rozróżnić użytkowników końcowych.

3 Włączanie logu WS

W standardowej konfiguracji WS nie tworzy logów w systemie plików. W celu zmiany tych domyślnych ustawień należy dokonać edycji pliku konfiguracyjnego …/Lev1/SASApp/WorkspaceServer/logconfig.xml (po uprzednim wykonaniu kopii bezpieczeństwa). Mechanizm log4sas, który bazuje na tym pliku konfiguracyjnym, umożliwia logowanie działania sesji SAS z dowolnie określonym poziomem szczegółowości. W naszych rozważaniach ograniczymy się do przykładu z poprzedniego rozdziału, w którym wspominaliśmy o potrzebie ustawienia logger-a Audit.Authentication. Oryginalny plik logconfig.xml należy uzupełnić o definicję tzw. appender-a plikowego, określającego plik i format komunikatów odkładanych w logu oraz definicję tzw. logger-a Audit.Authentication określającego poziom szczegółowości rejestrowanych przez niego komunikatów.

Należy przy tym pamiętać, aby użytkownik systemowy ‘grpusr’ miał uprawnienia pozwalające mu na tworzenie i modyfikację plików w katalogu wskazanym przez appender plikowy – jeżeli tych uprawnień nie będzie posiadać, sesje WS nie zostaną uruchomione, a użytkownicy aplikacji zobaczą błąd. Dodatkowo, jeżeli podczas edycji pliku logconfig.xml wkradnie nam się błąd (np. niedomknięty znacznik), sesja WS także nie zostanie uruchomiona.

Jeżeli WS jest skonfigurowany dla autentykacji tokenowej, uprawnienia do zapisu katalogu z logami powinien mieć użytkownik systemowy wskazywany przez atrybut ‘launch credential’ serwera. Jeżeli zaś WS jest skonfigurowany dla kont indywidualnych, uprawnienia do katalogu logów powinni mieć poszczególni użytkownicy indywidualni.

Poniżej przedstawiono definicje appender-a i logger-a, o które należy uzupełnić plik logconfig.xml dla WS.

```
...

   <appender class="RollingFileAppender" name="PlikLogu">

      <param name="Append" value="false"/>

      <param name="Unique" value="true"/>

      <param name="ImmediateFlush" value="true"/>

      <rollingPolicy class="TimeBasedRollingPolicy">

         <param name="FileNamePattern"
value="/opt/sasconf/Lev1/SASApp/WorkspaceServer/Logs/SASApp_WorkspaceServer_%d_%S{hostname}
_%S{pid}.log"/>

      </rollingPolicy>

      <layout>

         <param name="HeaderPattern" value="Host: '%S{hostname}', OS: '%S{os_family}',
Release: '%S{os_release}', SAS Version: '%S{sup_ver_long2}', Command: '%S{startup_cmd}'"/>

         <param name="ConversionPattern" value="%d %-5p [%t] %X{Client.ID}:%u - %m"/>

      </layout>

   </appender>

...

<logger name="Audit.Authentication">

   <appender-ref ref="PlikLogu"/>

   <level value="INFO"/>

</logger>

...
```