Metadane a SAS 4GL

Zespół Wsparcia Technicznego, SAS Institute Polska październik 2013

Szanowni Państwo!

Niniejszy dokument stanowi wprowadzenie do zagadnienia wykorzystania metadanych SAS-owych w trakcie tworzenia programów w SAS 4GL. Główne kwestie poruszane w tej publikacji to:

- Dostęp do zasobów zdefiniowanych w metadanych.
- Przegląd i modyfikacja metadanych z poziomu kodu w SAS 4GL.

Pełny opis zagadnień opisywanych w niniejszym opracowaniu znajduje się w dokumentach SAS® 9.4 Language Interfaces to Metadata, SAS® 9.4 Open Metadata Interface: Reference and Usage oraz SAS® 9.3 Metadata Model: Reference. Jeżeli posiadają Państwo starzą wersję oprogramowania SAS, odpowiedniki wymienionych dokumentów można znaleźć na stronie http://support.sas.com/documentation/index.html. Wszystkie przykłady zawarte w niniejszym dokumencie zostały sprawdzone dla SAS 9.2, SAS 9.3 oraz SAS 9.4.

Niniejszy dokument oraz inne informacje związane z tematem będą publikowane na naszych stronach WWW pod adresem http://www.sas.com/poland/services.

W razie dodatkowych pytań prosimy o kontakt z naszym Zespołem, telefonicznie na numer **(22) 560 4666** lub elektronicznie pod adresem support@spl.sas.com.

Prosimy o uwagi i spostrzeżenia, które pomogą nam w przygotowaniu następnych wersji tego dokumentu.

Zawsze służąc Państwu pomocą,
Zespół Pomocy Technicznej
Centrum Technologii SAS Polska

Spis treści

Spis treści ... 2
1 Wstęp ... 3
1.1. Struktura metadanych ... 4
1.2. Definiowanie połączenia do metadanych ... 5
1.3. Eksplorator metadanych w SAS Base ... 7
2 Dostęp do bibliotek zdefiniowanych w metadanych – Metadata Libname Engine ... 8
3 Aktualizacja biblioteki w metadanych – PROC METALIB ... 11
3.1 Składnia procedury ... 11
3.2 Aktualizacja definicji tabel w metadanych ... 12
3.3 Raporty oraz analiza wpływu ... 13
3.3.1 Domyślny raport ... 14
3.3.2 Analiza wpływu – raport podsumowujący. ... 15
3.3.3 Analiza wpływu – raport szczegółowy ... 16
4 Czynności administracyjne – PROC METAOPERATE ... 17
4.1 Składnia procedury ... 17
4.2 Przykłady ... 19
4.2.1 Wstrzymanie pracy serwera metadanych ... 19
4.2.2 Sprawdzanie stanu serwera metadanych ... 19
4.2.3 Przywrócenie działania serwera metadanych ... 20
4.2.4 Wykonanie backupu ... 20
4.2.5 Zmiana harmonogramu backup-ów. ... 20
5 Przeglądanie i modyfikacja metadanych – PROC METADATA ... 21
5.1 Składnia procedury ... 21
5.2 Sprawdzanie stanu serwera ... 22
5.3 Odczyt i manipulacja metadanymi. ... 23
5.3.1 Odczyt metadanych ... 24
5.3.2 Dodawanie metadanych - AddMetadata ... 30
5.3.3 Aktualizacja metadanych – UpdateMetadata ... 31
5.3.4 Usuwanie metadanych – DeleteMetadata ... 32
6 Dostęp do metadanych z poziomu DATA Step-u ... 33
6.1 Dostępne funkcje ... 33
6.2 Przykłady ... 34
6.2.1 Dodawanie użytkownika do grupy. ... 34
6.2.2 Rejestracja biblioteki SAS BASE w metadanych ... 35

1 Wstęp

Metadane to centralny element każdego większego systemu SAS-owego. Przechowują one informacje o konfiguracji środowiska (serwery SAS, bazy danych), użytkownikach (loginy, hasła, grupy, role), danych (biblioteki, tabele, procesy gotowe, raporty), prawach dostępu (uprawnienia do obiektu, listy ACL i ACT) i wiele innych. Praktycznie każdy element obecny w środowisku SAS-owych ma swoją reprezentację w metadanych.

Metadane są przechowywane w postaci bibliotek SAS-owych (silnik BASE), po jednej na każde repozytorium. Domyślnie przechowywane są one w folderze ***<SASConfig>\Lev1\SASMeta\MetadataServer\MetadataRepositories***. Za ich obsługę odpowiada Metadata Server.

Większość czynności związanych z metadanymi realizowanych jest za pomocą aplikacji klienckich. Operacje te dzielą się na administracyjno-utrzymaniowe oraz tworzenie nowych obiektów. Pierwsza grupa jest realizowana za pośrednictwem SAS Management Console. Pozostałe aplikacje klienckie, takie jak SAS Enterprise Guide, SAS Data Information Studio, SAS Information Map Studio, ale również aplikacje web-owe pozwalają głównie na tworzenie nowych obiektów.

Interfejsy graficzne mają jednak dwa podstawowe ograniczenia. Pierwszym z nich jest brak odzwierciedlenia wszystkich obiektów metadanowych. Przykładem takich obiektów są strony tworzone w aplikacji SASPortal, które nie są dostępne z poziomu SAS Management Console. Drugim ograniczeniem jest fakt, że w pewnych przypadkach interfejsy graficzne mogą być niewygodne. Jest to szczególnie widoczne w sytuacji, gdy chcemy utworzyć wiele podobnych do siebie obiektów, np. procesów gotowych.

Jeżeli chcemy dokonać zmian w niedostępnym obiekcie, bądź utworzyć całą serię obiektów bez konieczności żmudnego przechodzenia kolejnych kreatorów, możemy posłużyć się kodem SAS-owym. Działania te są możliwe przy użyciu **PROC METADATA**, na której skupimy się w niniejszym dokumencie.

UWAGA. Modyfikacja metadanych za pomocą PROC METADATA jest czynnością wysoce zaawansowaną i może w skrajnych przypadkach doprowadzić do nieodwracalnego uszkodzenia całego środowiska. Przed wykonaniem jakichkolwiek działań z użyciem tej procedury zaleca się przeprowadzenie backup-u metadanych.

1.1. Struktura metadanych

W metadanych przechowywane są wszystkie informacje niezbędne do działania środowiska SAS. Zakres typów informacji jest bardzo szeroki, od definicji serwerów przez uprawnienia użytkowników po definicje procesów gotowych. W rozdziale tym omówiony zostanie sposób, w jaki metadane są organizowane, aby umożliwić skuteczne gromadzić tak szeroki zakres informacji.

Każdy obiekt w metadanych ma przypisany typ, informujący, jakiego rodzaju informacja jest w nim przechowywana. Typ definiuje dodatkowo strukturę obiektu, na którą składają się 3 rodzaje informacji: atrybuty (ang. attribute), właściwości (ang. property) oraz asocjacje (ang. association). Atrybuty opisują cechy wybranego obiektu. Ich zestaw jest identyczny dla wszystkich obiektów danego typu. Właściwości stanowią rozszerzenie domyślnego zestawu atrybutów. Ich przeznaczeniem jest rozszerzenie domyślnego zestawu atrybutów. Asocjacje opisują powiązania obiektu metadanowego z innymi obiektami. Powiązania te są zorganizowane w postaci grup. Dostępne grupy asocjacji oraz ich kardynalność (ilość możliwych powiązań w danej grupie) wynikają bezpośrednio z typu obiektu.

Wszystkie typy wywodzą się od jednego typu bazowego, o nazwie Root. Definiuje on strukturę wspólną dla wszystkich obiektów w metadanych. W ramach obiektu Root zdefiniowane zostały następujące parametry (wartości obowiązkowe zostały pogrubione).

- **Id** – unikalny identyfikator obiektu. Identyfikator ma długość 17 znaków i składa się z dwóch 8-znakowych wartości oddzielonych kropką. Pierwsza wartość identyfikuje repozytorium, w którym przechowywany jest obiekt, druga natomiast służy do identyfikacji obiektu w ramach repozytorium. Przykładowa wartość identyfikatora to A5L9HV00.AT000001.
- **Name** – nazwa obiektu. Maksymalna długość nazwy wynosi 60 znaków.
- Desc – opis obiektu
- **MetadataCreated** – czas utworzenia obiektu
- **MetadataUpdated** – czas ostatniej modyfikacji obiektu
- ChangeState – wartość wykorzystywana do kontroli zmian, wykorzystywana przez SAS Data Integration Studio.
- LockedBy – wskazuje, który użytkownik aktualnie zablokował dany obiekt. Podobnie jak wartość ChangeStart jest ona wykorzystywana w mechanizmie kontroli zmian w Data Integration Studio.
- UsageVersion – identyfikuje wersję obiektu. Wartość ta jest pozwala na wykrycie sytuacji, kiedy obiekt został utworzony według starszej definicji obiektu. Przykładem takiej sytuacji mogą być procesy gotowe, których definicja uległa znaczącej zmianie w SAS 9.3. Wartość samej wersji zapisywana jest w postaci liczbowej według schematu MMMmmbbbb, gdzie M oznacza wersję główną (major), m – wersję poboczną (minor) a b – wersję kompilacji (build).

Oprócz wymienionych parametrów obiekt Root definiuje szereg asocjacji, z których najważniejsze to:

- AccessControls – pozwala na przypisanie do danego obiektu informacji o kontroli dostępu w postaci jednego lub więcej obiektów typu AccessControlEntry bądź AccessControlTemplate.

- Properties / Extensions – pozwala na definiowanie dodatkowych parametrów, niebędących częścią standardowego obiektu. Obiekty Properties dedykowane są dla twórców aplikacji, natomiast Extensions przeznaczone są bardziej dla użytkowników.
- Notes – zawiera wskazania na obiekt typu TextStore, przechowujący informację tekstową dla danego obiektu.

Typy obiektów dzielą się na dwie podstawowe grupy:

- Typy główne (PrimatyType) – ich zadaniem jest reprezentowanie głównych obiektów metadanowych bądź przechowywanie dodatkowych informacji dla takich obiektów. Obiekty typów głównych mogą być tworzone / modyfikowane i usuwane niezależnie od siebie. Dodatkowo każdy obiekt tego typu może mieć ustaloną własną konfigurację dostępu.
- Typy poboczne (SecondaryType) – ich przeznaczeniem jest przechowywanie dodatkowych informacji dla obiektów typów głównych. W odróżnieniu od obiektów typów głównych, obiekty typów pobocznych mają przypisany nadrzędny obiekt typu głównego, z którym są nierozerwalnie związane. Uprawnienia dostępu są zawsze dziedziczone z obiektu nadrzędnego.

Więcej informacji na temat struktury metadanych zawartych jest w dokumencie SAS® 9.3 Metadata Model: Reference.

1.2. Definiowanie połączenia do metadanych

Pierwszym krokiem, jaki należy wykonać chcąc uzyskać dostęp do metadanych z poziomu kodu w SAS 4GL jest definicja i nawiązanie połączenia do serwera metadanych (MetadataServer). Na definicję połączenia do serwera metadanych składa się poniższy zbiór opcji. Opcje te, ze względu na swoją funkcję dzielą się na 3 podstawowe grupy:

- Opcje lokalizacji – definiują lokalizację serwera metadanych oraz repozytorium, do którego chcemy się połączyć:
    - **METASERVER** – nazwa maszyny, na której znajduje się serwer metadanych lub jej adres IP.
    - **METAPORT** – port nasłuchu serwera metadanych (zazwyczaj 8561).
    - **METAREPOSITORY** – nazwa repozytorium w metadanych, do którego ma nastąpić połączenie (domyślnie Foundation).
- Opcje profilu – określają użytkownika, z którego prawami będą wykonywane operacje po nawiązaniu połączenia
    - **METAUSER** – nazwa użytkownika w metadanych.
    - **METAPASS** – hasło użytkownika w metadanych.
    - **METAPROFILE** – lokalizacja pliku XML zawierającego definicje połączeń do serwera metadanych.
    - **METACONNECT** – nazwa wykorzystywanego profilu z pliku wskazanego w opcji METAPROFILE.
    - **METASPN** – SPN dla zarządcy (principal), który uruchamia serwer metadanych. Opcja używana w przypadku korzystania ze zintegrowanego uwierzytelniania Windows (Integrated Windows

Authentication). Dozwolone formaty dla opcji to **usługa/nazwa_maszyny** oraz **usługa/nazwa_maszyny.company.com**.

- Opcje szyfrowania – za ich pomocą możemy określić, które dane mają być szyfrowane i którym algorytmem.
  - **METAENCRYPTLEVEL** – określa stopień szyfrowania komunikacji. Dostępne opcje to:
    - CREDENTIALS (domyślne) – szyfrowane są tylko informacje logowania.
    - EVERYTHING – szyfrowana jest cała komunikacja między serwerem metadanych a klientem.
  - **METAENCRYPTALG** – algorytm używany do szyfrowania komunikacji z serwerem metadanych. Dostępne algorytmy to: NONE (brak szyfrowania), RC2, RC4, DES, TripleDES, AES, SASProprietary(domyślny).

UWAGA. Lista algorytmów szyfrowania dostępnych w danym środowisku zależy od posiadanych licencji.

Wartości wymienionych opcji można ustalać w następujący sposób:

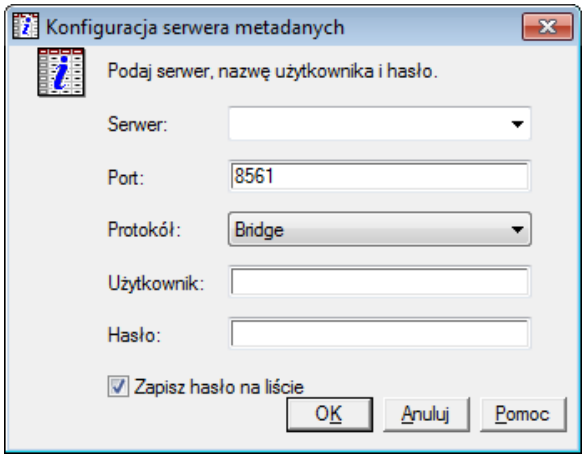
- Jako parametr sesji – w pliku konfiguracyjnym albo jako parametr uruchomienia.
- Jako opcję w czasie trwania danej sesji – instrukcja **options**.

Aktualne wartości wszystkich opcji związanych z połączeniem do serwera metadanych można uzyskać wykonując poniższe polecenie:

```
proc options group=meta;
run;

Wynik:

METAAUTORESOURCES=
                    Specifies the metadata resources that are assigned when SAS starts.
METACONNECT=        Specifies the profile from the metadata user connection profiles that
                    is used to connect to the SAS Metadata Server.
METAENCRYPTALG=SASPROPRIETARY
                    Specifies the type of encryption to use to communicate with the SAS
                    Metadata Server.
METAENCRYPTLEVEL=CREDENTIALS
                    Specifies the level of encryption that is used to communicate with the
                    SAS Metadata Server.
METAID=             Specifies the ID of the SAS Metadata Server.
METAPASS=XXXXXXXX   Specifies the SAS Metadata Server password.
METAPORT=8561       Specifies the TCP port for the SAS Metadata Server.
METAPROFILE=        Specifies the XML document that contains SAS Metadata Server user
                    connection profiles.
 METAPROTOCOL=BRIDGE
                    Specifies the network profile to use to connect to the SAS Metadata

Server.
METAREPOSITORY=Foundation
                    Specifies the name of the SAS Metadata Server Repository.
METASERVER=         Specifies the host name or address of the SAS Metadata Server.
METASPN=            Specifies the service principal name (SPN) for the SAS Metadata Server.
METAUSER=           Specifies the user ID that is used to connect to the SAS Metadata
                    Server.
```

W standardowym przypadku do połączenia z serwerem metadanych wystarczy podanie parametrów METAHOST, METAPORT, METAUSER oraz METAPASS. Pozostałe parametry są wykorzystywane jedynie w

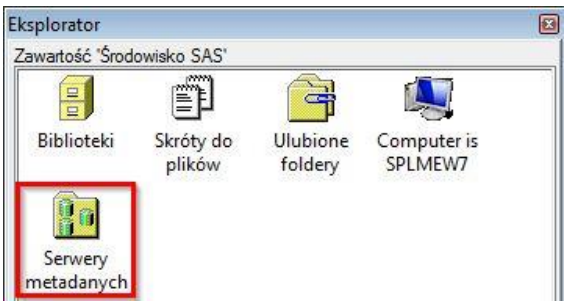
specyficznych przypadkach. Jeżeli korzystamy z sesji graficznej, np. pisząc kod w SAS Base bądź SAS Enterprise Guide, nie ma potrzeby wcześniejszego definiowania wymienionych opcji. W przypadku braku wartości któregoś z parametrów wyświetlone zostanie okno z prośbą o uzupełnienie informacji. Przykład takiego okna zamieszczony został na poniższym rysunku.



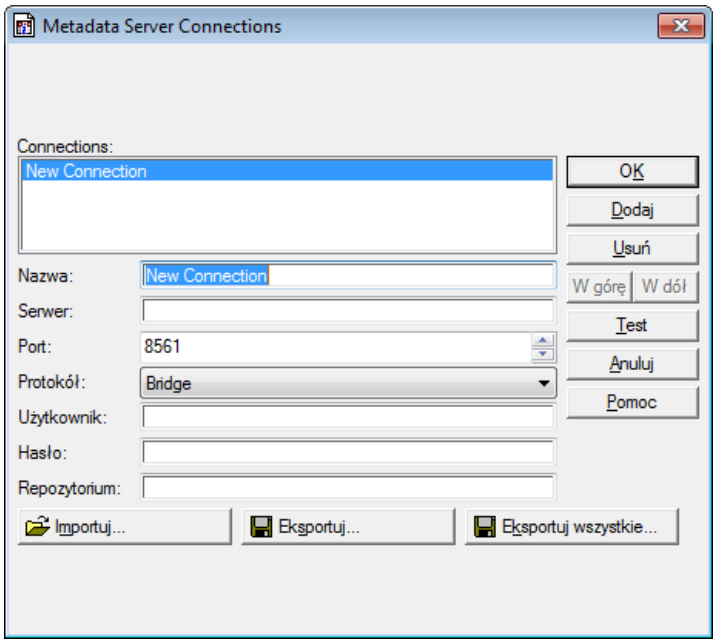
W przypadku przetwarzania wsadowego opisywana możliwość nie jest dostępna i konieczne jest podanie pełnej definicji połączenia do serwera metadanych w kodzie procesu

1.3. Eksplorator metadanych w SAS Base

SAS Base posiada wbudowane narzędzie, pozwalające na przegląd i edycję metadanych. W celu włączenia tego narzędzia należy kliknąć okno Eksploratora a następnie wybrać ***Narzędzia -> Opcje -> Eksplorator***. W wyświetlonym oknie w zakładce Ogólne należy zaznaczyć komponent ***Serwery metadanych***. Jeżeli planujemy modyfikować metadane należy odznaczyć opcję ***Tryb przeglądania metadanych***. W wyniku zmian na najwyższym poziomie eksploratora powinna pojawić się nowa pozycja, zaznaczona poniższym rysunku.



Po przejściu do zaznaczonego folderu należy utworzyć nowy profil połączeniowy do serwera metadanych. W tym celu należy kliknąć prawym przyciskiem myszy (PPM) w obszarze okna eksploratora i wybrać ***Nowy…*** . Poniżej przedstawione zostało okno tworzenia nowego połączenia.



Po zdefiniowaniu połączenia w oknie eksploratora pojawi się ikona reprezentująca połączenie. Po wejściu w połączenie i ewentualnym wyborze repozytorium metadanych uzyskujemy dostęp do wszystkich obiektów metadanowych, pogrupowanych według typu.

2 Dostęp do bibliotek zdefiniowanych w metadanych – Metadata Libname Engine

Jedną z głównych korzyści płynących z wykorzystania metadanych jest łatwy dostęp do wszystkich bibliotek w nich przechowywanych. Dzięki temu użytkownicy nie muszą za każdym uruchomieniem klienta definiować od nowa połączenia do źródeł danych. Udogodnienia te są również dostępne z poziomu kodu w SAS 4GL, poprzez wykorzystanie mechanizmu Metadata Libname Engine (MLE). Jest to specjalny silnik w instrukcji libname, pozwalający na odczyt definicji biblioteki.

Proces podłączania biblioteki oraz jej późniejsze używanie przebiega analogicznie jak w przypadku każdego innego typu biblioteki. Wykorzystanie mechanizmu MLE daje jednak szereg udogodnień:

- **Mniej opcji połączenia** – w przypadku podłączania biblioteki bez użycia metadanych konieczne jest zdefiniowanie wszystkich opcji połączenia, które mają mieć inne wartości niż domyślnie. W przypadku połączeń do baz danych takich parametrów może być bardzo dużo. Jeżeli korzystamy z definicji zlokalizowanej w metadanych, wystarczy znać nazwę, jaka została nadana tej bibliotece na serwerze. Definicja biblioteki przechowywana w metadanych zawiera wszystkie opcje potrzebne do właściwego połączenia z biblioteką i nie ma potrzeby ponownego ich definiowania.
- **Kontrola dostępu** – aby móc przeczytać definicję biblioteki i nawiązać z nią połączenie użytkownik musi przejść przez proces autentykacji i autoryzacji. Dzięki temu dostęp do biblioteki odbywa się zgodnie z systemem uprawnień zdefiniowanym przez administratora

- **Więcej informacji o kolumnach** – część źródeł danych nie przechowuje części informacji o kolumnach dostępnych w SAS., takich jak formaty, informaty czy etykiety. Definicja biblioteki w metadanych zawsze zawiera takie informacje, niezależnie od ostatecznego źródła danych.
- **Nie trzeba pamiętać haseł** – dostęp do zewnętrznych baz danych wymaga zazwyczaj podania loginu i hasła do konta na tej bazie danych. W przypadku korzystania z metadanych SAS jest w stanie automatycznie znaleźć te informacje, o ile są one dostępne dla użytkownika, którym łączymy się do metadanych.

Składnia instrukcji LIBNAME korzystającej z mechanizmu MLE jest następująca:

```
LIBNAME <libref> META
    LIBID="identyfikator_biblioteki" || LIBRARY="nazwa_biblioteki" || LIBURI="URI-
biblioteki"
    <opcje_połączenia_do_serwera>
    <METAOUT=ALL | DATA | DATAREG | META>
;
```

Wartości dla poszczególnych opcji są następujące:

- Opcje wyboru biblioteki – biblioteka może zostać zdefiniowana na 1 z 3 sposobów:
  - LIBID – identyfikator obiektu biblioteki w metadanych. Jest to 8 lub 17-znakowy ciąg znaków identyfikujący obiekt biblioteki w metadanych. Przykładowe wartości to libid="AW000002" oraz libid="A57DQR88.AW000002".
  - LIBRARY – nazwa biblioteki widziana z poziomu SAS Management Console. Przykładowa wartość: library=test001;
  - LIBURI – URI do obiektu reprezentującego bibliotekę w metadanych. Dostępne format URI to:
    - Długi identyfikator obiektu, np. liburi="A57DQR88.AW000002"
    - <typ_obiektu>/<długi_identyfikator>, np. Liburi="SASLibrary/A57DQR88.AW000002"
    - <typ_obiektu>?<parametry_przeszukiwania>, np. liburi="SASLibrary?@libref='mylib'"
- Opcje połączenia
  - METASERVER – nazwa serwera metadanych lub jego adres IP, np. Metaserver="localhost"
  - METAPORT – port nasłuchu serwera metadanych, np. Metaport="8561"
  - METAUSER – nazwa użytkownika, np. Metauser="sasadm@saspw"
  - METAPASS – hasło użytkownika, np. Metapass="admin123 "
- METAOUT – opcja typu operacji wykonywanych na bibliotece
  - ALL (domyślny) – dozwolony jest odczyt, tworzenie, modyfikacja i usuwanie obserwacji w tabelach zdefiniowanych w metadanych. Nie ma możliwości usuwania ani tworzenia fizycznych tabel.
  - DATA – dozwolony jest odczyt, tworzenie, modyfikacja i usuwanie fizycznych tabel, nawet tych niezdefiniowanych w metadanych.
  - DATAREG – dozwolony jest odczyt, modyfikacja i usuwanie tabel zdefiniowanych w metadanych. Wartość ta działa jak ALL z dodatkową możliwością tworzenia nowych tabel
  - META – dozwolony jest odczyt informacji o tabelach z metadanych. Użytkownik nie ma możliwości modyfikacji obserwacji.

Przykład użycia mechanizmu Metadata Libname Engine:

```
options metaserver="localhost"
       metaport=8561
       metauser="sasdemo"
       metapass="haslo1";

libname samples meta library="samples";
libname oracle  meta library="orcl";

Wynik:

NOTE: Libref SAMPLES was successfully assigned as follows:
      Engine:        META
      Physical Name: C:\SAS\libs\samples

NOTE: Libref ORACLE was successfully assigned as follows:
      Engine:        META
      Physical Name: ORCL
```

3 Aktualizacja biblioteki w metadanych – PROC METALIB

Zadaniem PROC METALIB jest aktualizacja informacji o bibliotece przechowywanych w metadanych, tak, aby odpowiadały one faktycznym danym przechowywanym na dysku bądź w bazie danych. Oprócz tego procedura ta pozwala na generowanie raportów o zawartości biblioteki oraz przeprowadzenie analizy wpływu (impact analysis) dla konkretnych tabel.

3.1 Składnia procedury

PROC METALIB OMR=(identyfikator biblioteki) <EXCLUDE = (lista tabeli)> | <SELECT = (lista tabeli)> <FOLDER=(ścieżka folderu)> | <FOLDERID = (identyfikator folderu)> <IMPACT_LIMIT = n> <NOEXEC> <PREFIX = (treść prefiksu)> <REPORT = (typ raportu)> <UPDATE_RULE> = (reguła aktualizacji)

Do podstawowego działania procedury wymagany jest jedynie parametr OMR, zawierający informację o bibliotece, która będzie badana. Pozostałe parametry są opcjonalne. Poniżej znajduje się dokładny opis poszczególnych parametrów:

- OMR – wskazuje obiekt biblioteki w metadanych, który będzie sprawdzany w ramach procedury. Obiekt może zostać zdefiniowane na 1 z 3 sposobów:
    - Identyfikator biblioteki – LIBID="id", gdzie ID jest 8-znakowym identyfikatorem w ramach obiektów typu SASLibrary.
    - Nazwa biblioteki – LIBRARY="nazwa"
    - URI obiektu – LIBURI="wyrażenie URI". Samo wyrażenie URI może przyjąć formy opisywane w rozdziale 2.
- SELECT lub EXCLUDE – wskazuje listę tabeli, które będą sprawdzane (SELECT) bądź pominięte (EXCLUDE) podczas działania procedury METALIB. Poszczególne nazwy tabeli należy podawać w cudzysłowach i oddzielać znakiem spacji.
- FOLDER lub FOLDERID – wskazuje lokalizację w metadanych (FOLDER) bądź identyfikator folderu (FOLDERID), w którym zapisane mają być obiekty opisujące tabele w biblioteki.
- PREFIX – określa przedrostek, jaki będzie dodawany do nazwy obiektu w metadanych dla danej tabeli.
- UPDATE_RULE – definiuje zachowanie procedury podczas znalezienia rozbieżności pomiędzy fizyczną tabelą a jej opisem w metadanych. Możliwe wartości to:
    - DELETE – włącza usuwanie definicji w metadanych, jeżeli tabela, do której się obwołuje, już nie istnieje. Oprócz tego włącza usuwanie duplikatów w przypadku, jeżeli jednej tabeli fizycznej odpowiada kilka obiektów w metadanych.
    - NODELDUP – wyłącza usuwanie duplikatów definicji w metadanych. Wymaga ustawienia opcji DELETE.
    - NOADD – wyłącza tworzenie obiektów w metadanych dla tabel, które nie mają swojego opisu w metadanych.
    - NOUPDATE – wyłącza aktualizację informacji o tabeli.

Opcja pozwala na jednoczesne podanie kilku z wymienionych powyżej opcji.

- REPORT – powoduje wygenerowanie dodatkowego raportu SAS-owego, podsumowującego zmiany dokonane w metadanych. Możliwe jest zdefiniowanie jednego z 2 typów raportów:
    - SUMMARY – raport podsumowujący, zawiera informacje o zmianach dokonanych w metadanych dla przetwarzanych tabel
    - DETAIL – raport szczegółowy, zawiera wszystkie informacje zawarte w raporcie podsumowującym oraz dodatkowo zwraca listę prac (job) i transformacji (transformation), które mogą zostać dotknięte przez daną modyfikację. Dla uzyskania pełnych wyników wymagane jest zdefiniowanie opcji IMPACT_LIMIT

Dla obu typów raportów można dodatkowo ustawić opcję MATCHING, włączającą wyświetlanie informacji o tabelach, które nie uległy modyfikacji.

- NOEXEC – powoduje wyłączenie modyfikacji metadanych. Opcja szczególnie przydatna w połączeniu z opcjami REPORT oraz IMPACT_LIMIT.
- IMPACT_LIMIT – określa maksymalną liczbę prac (job) lub transformacji (transformation). Jeżeli liczba ta zostanie przekroczona, zmiany nie zostaną wprowadzone.

Opcja IMPACT_LIMIT=0 w połączeniu z opcją REPORT pozwala na przeprowadzenie analizy wpływu (impact analysis) zmiany danej tabeli. W zależności od wartości opcji REPORT można uzyskać następujące informacje:

- REPORT (TYPE=SUMMARY) – które tabele są używane przez prace bądź transformacje.
- REPORT (TYPE=DETAIL) – zawiera informacje które prace / transformacje korzystają z danej tabeli

3.2 Aktualizacja definicji tabel w metadanych

Procedura aktualizacji biblioteki polega na połączeniu się do źródła danych, wykorzystywanego przez daną bibliotekę a następnie porównaniu informacji o poszczególnych tabelach z ich reprezentacjami w metadanych. W przypadku znalezienia rozbieżności procedura METALIB podejmuje odpowiednie działania w celu jej usunięcia. Działaniami tymi są:

- utworzenie obiektu tabeli w metadanych w przypadku, gdy tabela istnieje fizycznie w bibliotece ale nie ma swojego odpowiednika w metadanych (o ile nie została ustawiona opcja UPDATE_RULE=(NOADD))
- aktualizacja obiektu tabeli w metadanych, w przypadku gdy zostanie wykryta rozbieżność (o ile nie została ustawiona opcja UPDATE_RULE=(NOUPDATE))
- usuwanie definicji metadanowych opisujących tabele, które już fizycznie nie istnieją (o ile została ustawiona opcja UPDATE_RULE=(DELETE)) oraz osuwanie duplikatów w metadanych (o ile nie została ustawiona opcja UPDATE_RULE=(DELETE NODELDUP))

Domyślnie procedura METALIB wykonuje zatem dodawanie i aktualizację definicji metadanowych dla tabeli.

Poniżej zamieszczone zostały przykłady użycia proc metalib.

```
proc metalib;
    OMR(library="samples");
run;

Wynik:

NOTE: A total of 5 tables were analyzed for library "samples".
NOTE: Metadata for 0 tables was updated.
NOTE: Metadata for 2 tables was added.
NOTE: Metadata for 3 tables matched the data sources.
NOTE: 0 other tables were not processed due to error or UPDATE_RULE.
NOTE: PROCEDURE METALIB used (Total process time):
      real time           0.13 seconds
      cpu time            0.04 seconds
```

Przykład 1 PROC METALIB - domyślne uruchomienie

PROC METALIB nie działa dla bibliotek pre-assigned. Próba dostępu do takiej biblioteki generuje następujący błąd: ERROR: Libref <nazwa_biblioteki> is not assigned."

```
proc metalib;
    OMR(libid="AZ0001K6");
    FOLDER="/Shared Data/02 Tables/samples";
    SELECT ("class" "cars");
run;

Wynik:

NOTE: A total of 2 tables were analyzed for library "samples".
NOTE: Metadata for 0 tables was updated.
NOTE: Metadata for 1 tables was added.
NOTE: Metadata for 1 tables matched the data sources.
NOTE: 0 tables listed in the SELECT or EXCLUDE statement were not found in either the
metadata or
      the data source.
NOTE: 0 other tables were not processed due to error or UPDATE_RULE.
NOTE: PROCEDURE METALIB used (Total process time):
      real time           0.08 seconds
      cpu time            0.06 seconds
```

Przykład 2 PROC METALIB – podzbiór tabel oraz inny katalog obiektów w metadanych

UWAGA. Domyślnie obiekty tabel tworzone przez PROC METALIB zapisywane są w tym samym folderze co obiekt reprezentujący bibliotekę.

3.3 Raporty oraz analiza wpływu

Procedura METALIB oprócz samej modyfikacji metadanych jest w stanie wygenerować raport opisujący znalezione różnice. W przypadku ustawienia opcji IMPACT_LIMIT raport może zawierać dodatkowo informacje o pracach i transformacjach, które wykorzystują daną tabelę.

W następnych punktach zamieszczone zostały przykładowe raporty, możliwe do uzyskania przy użyciu procedury METALIB.

3.3.1 Domyślny raport

Kod:

```
proc metalib;
    omr(library=samples);
    FOLDER="/Shared Data/02 Tables/samples";
    NOEXEC;
    REPORT:
run;
```

Wynik:

The SAS System

The METALIB Procedure

Detail Report of Potential Changes for Library SAMPLES
Repository Foundation
27SEP2013

Metadata Summary Statistics	
Total tables analyzed	7
Tables to be Updated	1
Tables to be Added	1
Tables matching data source	5
Tables not processed	0

Tables to be Updated							
Table			Updates				
Metadata Name	Metadata ID	SAS Name	Metadata Name	Metadata ID	SAS Name	Metadata Type	Change
MODIFIED_TABLE	A5L9HV00.B70001L6	MODIFIED_TABLE	PersonName	A5L9HV00.B800021Z	PersonName	Column	SASColumnLength
							ColumnLength

Tables to be Added		
Metadata Name	Metadata ID	SAS Name
		NEW_TABLE

3.3.2 Analiza wpływu – raport podsumowujący.

Kod:

```
proc metalib;
    omr(library=samples);
    FOLDER="/Shared Data/02 Tables/samples";
    NOEXEC;
    IMPACT_LIMIT=0;
    REPORT(MATCHING);
run;
```

Wynik:

The METALIB Procedure

Summary Report of Potential Changes for Library SAMPLES
Repository Foundation
20SEP2013

Metadata Summary Statistics	
Total tables analyzed	5
Tables not processed due to large impact	1
Tables to be Updated	0
Tables to be Added	0
Tables matching data source	4
Tables not processed	0

Tables Exceeding Impact Limit of 0			
Metadata Name	Metadata ID	SAS Name	Total Impact
CARS	A5L9HV00.B70001KP	CARS	2

Tables matching data source		
Metadata Name	Metadata ID	SAS Name
MONTH	A5L9HV00.B70001L4	MONTH
CLASS	A5L9HV00.B70001KZ	CLASS
PRDSALE	A5L9HV00.B70001KT	PRDSALE
HEART	A5L9HV00.B70001KR	HEART

3.3.3 Analiza wpływu – raport szczegółowy

Kod:

```
proc metalib;
    omr(library=samples);
    FOLDER="/Shared Data/02 Tables/samples";
    NOEXEC;
    IMPACT_LIMIT=0;
    REPORT(TYPE=DETAIL MATCHING);
run;
```

Wynik:

The METALIB Procedure

Detail Report of Potential Changes for Library SAMPLES
Repository Foundation
20SEP2013

Metadata Summary Statistics	
Total tables analyzed	5
Tables not processed due to large impact	1
Tables to be Updated	0
Tables to be Added	0
Tables matching data source	4
Tables not processed	0

Tables Exceeding Impact Limit of 0			
Metadata Name	Metadata ID	SAS Name	Total Impact
CARS	A5L9HV00.B70001KP	CARS	2

Potential-Impact Analysis						
Table			Impact			
Metadata Name	Metadata ID	Action	Metadata Name	Metadata ID	Metadata Type	Transformation Role
CARS	A5L9HV00.B70001KP	Limit Exceeded	sample_job_01	A5L9HV00.BV0000S3	Job	
			sample_job_02	A5L9HV00.BV0000S4	Job	

Tables matching data source		
Metadata Name	Metadata ID	SAS Name
MONTH	A5L9HV00.B70001L4	MONTH
CLASS	A5L9HV00.B70001KZ	CLASS
PRDSALE	A5L9HV00.B70001KT	PRDSALE
HEART	A5L9HV00.B70001KR	HEART

4 Czynności administracyjne – PROC METAOPERATE

Procedura METAOPERATE została stworzona z myślą o wsadowym wykonywaniu czynności administracyjnych związanych z serwerem metadanych. Za jej pomocą można wykonać następujące operacje:

- Usuwanie, czyszczenie oraz wyrejestrowywania repozytoriów.
- Wstrzymywanie i wznawianie pracy serwera.
- Odświeżanie serwera metadanych w celu
    - Odzyskania pamięci,
    - Przeładowania informacji związanych z autoryzacją użytkowników,
    - Włączenie bądź wyłączenia zbierania logów wydajnościowych (ARM).
- Zatrzymania bądź pobrania informacji o stanie serwera metadanych
- Wykonywanie kopii zapasowej
- Zmiana konfiguracji i harmonogramu wykonywania kopii zapasowej
- Przywracanie serwera metadanych z kopii zapasowej

4.1 Składnia procedury

Składnia procedury METAOPERATE jest następująca:

PROC METAOPERATE ACTION=<akcja> <opcje dodatkowe>

Dostępne akcje to:

- STATUS – zwraca informację o stanie serwera metadanych,
- STOP – zatrzymuje całkowicie serwer metadanych. Po wykonaniu tej operacji konieczne jest ręczne uruchomienie serwera z poziomu systemu operacyjnego,
- PAUSE – powoduje wstrzymanie pracy serwera metadanych. W stanie tym serwer w dalszym ciągu jest uruchomiony lecz jego funkcjonalność jest ograniczona. Dokładny zakres dozwolonych operacji można wskazać w opcjach dodatkowych w parametrze SERVER STATE. Dostępne są następujące stany:
    - ADMIN – dostęp do serwera mają tylko administratorzy
    - READONLY – użytkownicy wciąż mają dostęp do serwera, lecz w trybie tylko do odczytu
    - OFFLINE – serwer niedostępny dla nikogo, możliwe są jedynie operacje administracyjne, np. tworzenie backup-ów.
- RESUME – przywraca serwer metadanych ze stanu wstrzymania do normalnego trybu działania.
- REFRESH – powoduje wstrzymanie pracy serwera metadanych lub wybranego repozytorium a następnie przywrócenie go do normalnego trybu działania. Akcja ta ma na celu odzyskanie pamięci zajętej przez serwer metadanych oraz odświeżenie reguł autoryzacji użytkowników.
- EMPTY – usuwa zawartość repozytorium wskazanego przez parametr REPOSITORY w konfiguracji połączenia bądź przez opcję systemową METAREPOSITORY.
- DELETE – usuwa zawartość oraz definicję repozytorium wskazanego przez parametr REPOSITORY lub opcję METAREPOSITORY.

- UNREGISTER – wyrejestrowuje repozytorium bez usuwania jego zawartości.

W sekcji opcji dodatkowych można zdefiniować następujące wartości:

- NOAUTOPAUSE – wyłącza automatyczne wstrzymywanie i przywracanie serwera metadanych podczas wykonywania operacji DELETE, EMPTY, UNREGISTER oraz REFRESH (w specyficznych przypadkach). Jeżeli opcja nie zostanie ustawiona, w trakcie wykonywania wymienionych operacji wstrzymana zostanie praca wszystkich repozytoriów, nawet jeżeli nie będą one modyfikowane.
- NOCLUSTER – opcja używana w przypadku wykonywania operacji STOP na sklastrowanym serwerze metadanych (od SAS 9.4). Powoduje ona wyłączenie domyślnej akcji polegającej na zatrzymywaniu wszystkich serwerów w klastrze. W przypadku korzystania z pojedynczego serwera metadanych, opcja ta jest ignorowana.
- NOREDIRECT – opcja używana w przypadku wykonywania operacji STATUS na sklastrowanym serwerze metadanych (od SAS 9.4). Powoduje ominięcie mechanizmu load-balancera w celu uzyskania informacji o konkretnym węźle w klastrze. W przypadku korzystania z pojedynczego serwera metadanych, opcja ta jest ignorowana.
- OUT – pozwala na zapis wyników akcji STATUS do wskazanego zbioru SAS-owego
- OPTIONS=”XML z opcjami” – pozwala na definiowanie dodatkowych opcji dla wykonywanych akcji. Wartości poszczególnych opcji muszą być zdefiniowane postaci dokumentu XML. Poniżej wymienione zostały najczęściej stosowane opcje:
    - BACKUP – pozwala na wykonanie backupu ad hoc podczas wykonywania operacji REFRESH.
    - BACKUPCONFIGURATION – pozwala na zmianę parametrów wykonywania backupów, jak katalog zapisu, czas przechowywania backup-ów oraz działanie harmonogramu backupów.
    - FORCE – pozwala na wykonanie operacji PAUSE oraz RESUME, w przypadku zawieszenia się procesu odbudowy metadanych.
    - PAUSECOMMENT – ustawia tekst komentarza w wyświetlany użytkownikom przy próbie dostępu do wstrzymanego serwera metadanych
    - SERVER STATE – opcja używana z akcją PAUSE; pozwala na ustawienie w jakim stanie ma znajdować się wstrzymany serwer. Dostępne są następujące stany:
        - PAUSE – wartość domyślna, dostęp tylko dla administratorów; możliwe tylko wykonanie i przywrócenie danych z backupu.
        - ADMIN – dostęp do środowiska mają tylko administratorzy
        - READONLY – użytkownicy nadal mogą się logować, jednak dostęp do metadanych odbywa się na zasadzie tylko-do-odczytu.

Pełna lista dostępnych opcji oraz ich parametrów znajduje się w dokumencie SAS® 9.4 Language Interfaces to Metadata.

4.2 Przykłady

W poniższych punktach przedstawione zostały przykładowe kody wykonujące wybrane akcje procedury METAOPERATE.

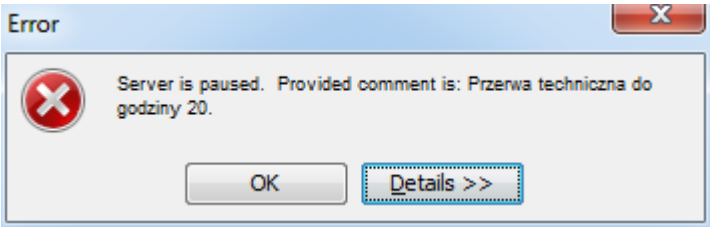
4.2.1 Wstrzymanie pracy serwera metadanych

Kod:

```
proc metaoperate
    action=pause
    options='
        <Server State="ADMIN"/>
        <PauseComment>Przerwa techniczna do godziny 20.</PauseComment>'
    ;
run;
```

Wynik:

Wszyscy użytkownicy zostają wylogowani z system. Przy próbie ponownego zalogowania osoby nie będące administratorami otrzymują następujący komunikat:



W zależności od aplikacji treść komunikatu może się nieznacznie różnić.

4.2.2 Sprawdzanie stanu serwera metadanych

Kod:

```
proc metaoperate
    action=status;
run;
```

Wynik:

```
NOTE: Server localhost SAS Version is 9.03.01M2P081512.
NOTE: Server localhost SAS Long Version is 9.03.01M2P08152012.
NOTE: Server localhost Operating System is X64_7PRO.
NOTE: Server localhost Operating System Family is WIN.
NOTE: Server localhost Operating System Version is Service Pack 1.
NOTE: Server localhost Client is sasadm@saspw.
NOTE: Server localhost Metadata Model is Version 12.05.
NOTE: Server localhost is PAUSED on 27 września 2013 13:25:58.
```

4.2.3 Przywrócenie działania serwera metadanych

Kod:

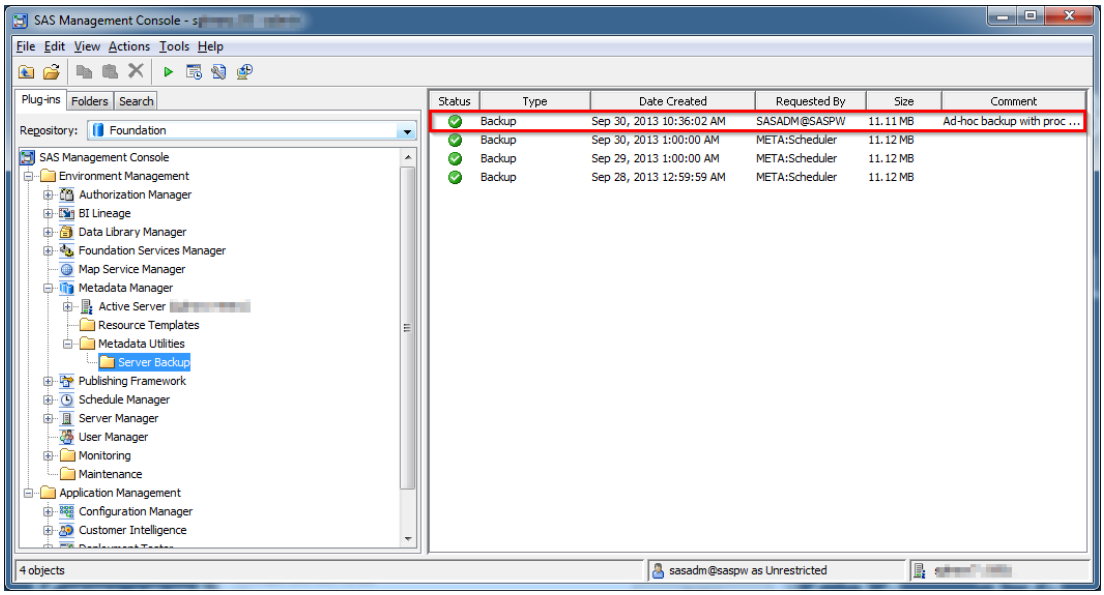
```
proc metaoperate
    action=resume;
run;
```

4.2.4 Wykonanie backupu

Kod:

```
proc metaoperate
    action=refresh
    options='<Backup reorg="y"/ comment="Ad-hoc backup with proc metaoperate">';
run;
```

Wynik:

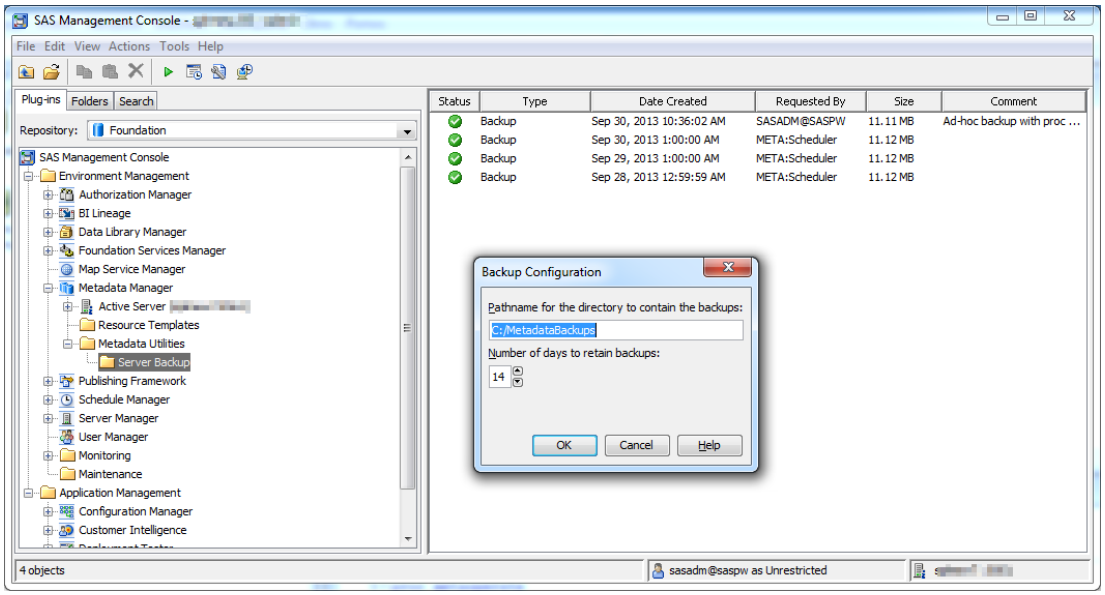


4.2.5 Zmiana harmonogramu backup-ów.

Kod:

```
proc metaoperate
    action=refresh
    options='<BackupConfiguration BackupLocation="C:\MetadataBackups"
RunScheduledBackups="y" DaysToRetainBackups="14"/>';
run;
```

Wynik:



5 Przeglądanie i modyfikacja metadanych – PROC METADATA

Procedura METADATA umożliwia wysyłanie szeregu poleceń do serwera metadanych z poziomu kodu SAS-owego. Polecenia te dzielą się na 2 podstawowe grupy:

- odczyt i modyfikacja metadanych – wykorzystanie metody DoRequest interfejsu SAS Open Metadata Interface (IOMI)
- sprawdzanie stanu serwera – wykorzystanie metody Status interfejsu IServer

5.1 Składnia procedury

PROC METADATA METHOD=STATUS | DOREQUEST IN="xml" | fileref <opcje dodatkowe>

Znaczenie poszczególnych parametrów oraz dostępne opcje wymienione zostały poniżej.

- METHOD – wybór typu polecenia, jakie ma zostać wysłane do serwera metadanych
  - DOREQUEST – wartość domyślna; zapytanie dotyczące metadanych, zgodne ze specyfikacją interfejsu IOMI
  - STATUS – zapytanie o stan serwera metadanych
- IN – treść polecenia wysyłanego do serwera metadanych. Treść ta może zostać podana na 2 sposoby:
  - „xml" – bezpośredni zapis treści wysyłanego zapytania w formacie xml
  - fileref – referencja do pliku zawierającego treść wysyłanego zapytania, również w formacie xml
- HEADER – definiuje czy odpowiedź serwera metadanych ma zawierać nagłówki zwracanego dokumentu XML. Opcja ta przydaje się w przypadku automatycznego przetwarzania odpowiedzi.
  - NONE – brak informacji nagłówkowych
  - SIMPLE – wartość domyślna; zwraca informacje o wersji użytego standardu xml
  - FULL – nagłówek odpowiedzi zawiera dodatkową informację o użytym standardzie kodowania znaków (ang. encoding), np. wlatin2.

- OUT – wskazuje fileref do pliku, w którym ma zostać zapisana odpowiedź otrzymana z serwera metadanych. Jeżeli opcja ta zostanie pominięta, odpowiedź jest umieszczana w logu.
- VERBOSE – wypisuje do logu treść zapytania wysłanego do serwera metadanych
- NOREDIRECT - opcja używana w przypadku wykonywania operacji STATUS na sklastrowanym serwerze metadanych (od SAS 9.4). Powoduje ominięcie mechanizmu load-balancera w celu uzyskania informacji o konkretnym węźle w klastrze. W przypadku korzystania z pojedynczego serwera metadanych, opcja ta jest ignorowana.

5.2 Sprawdzanie stanu serwera

Procedura METADATA, podobnie jak METAOPERATE pozwala na pobranie informacji o stanie serwera metadanych. Procedura METADATA pozwala na uzyskanie większej ilości informacji o serwerze, gdyż pozwala na definiowanie dowolnych parametrów metody Status, co nie jest możliwe przy użyciu PROC METAOPERATE. Dodatkowo informacje zwracane są w postaci dokumentu XML, co może być przydatne do przetwarzania przez zewnętrzne programy bądź w celach raportowych. Zaletą PROC METAOPERATE jest natomiast prostsza składnia, nie wymagająca definiowania dokumentu XML z treścią zapytania.

Przykładowe zapytania:

```
/* Sprawdzenie podstawowych informacji o stanie serwera*/
proc metadata
    method=status
    in=' ';
run;

/* Sprawdzenie konfiguracji backupowania serwera*/
proc metadata
    method=status
    in='<MetadataServerBackupConfiguration/>';
run;

/* sprawdzenie innych parametrów serwera metadanych */
proc metadata
    method=status
    in='<OMA JournalState=""/>
        <OMA JournalHistoricalData=""/>
        <OMA JournalPath=""/>
        <OMA ServerStartPath=""/>';
run;
```

Wyniki:

```
<ModelVersion>12.05</ModelVersion>
<PlatformVersion>9.3.0.0</PlatformVersion>
<ServerState>ONLINE</ServerState>
<PauseComment/>
<ServerLocale>pl_PL</ServerLocale>

<MetadataServerBackupConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xsi:noNamespaceSchemaLocation="MetadataServerBackupConfiguration.xsd"><MetadataSe
rverGUID="E76CBD5C-1225-4425-AAE0-A7755CAA37B9"/>
<Schedules><Schedule Event="Backup" Weekday1="0100"Weekday2="0100R" Weekday3="0100"
Weekday4="0100" Weekday5="0100" Weekday6="0100"Weekday7="0100"/></Schedules>
<BackupConfiguration BackupLocation="C:/MetadataBackups"DaysToRetainBackups="14"
RunScheduledBackups="Y"/>
</MetadataServerBackupConfiguration>
```

```
<OMA JournalState="IDLE"/><OMA JournalHistoricalData="29690"/><OMA JournalPath=""/><OMA
ServerStartPath=""/>
```

Uzyskanie informacji o stanie serwera jest również możliwe przy użyciu METHOD=DOREQUEST, wymaga jednak pewnej modyfikacji wysyłanego zapytania. Przykładowe zapytania o stan zostało przedstawione na poniższym przykładzie.

Kod:

```
proc metadata
    in='<Status><Metadata/></Status>';
run;
```

Wynik:

```
<Status>
    <Metadata>
        <ModelVersion>12.05</ModelVersion>
        <PlatformVersion>9.3.0.0</PlatformVersion>
        <ServerState>ONLINE</ServerState>
        <PauseComment/>
        <ServerLocale>pl_PL</ServerLocale>
    </Metadata>
</Status>
```

Pełna informacja na temat informacji możliwych do uzyskania za pomocą PROC METADATA z opcją METHOD=STATUS znajduje się w dokumencie **SAS® 9.4 Open Metadata Interface: Reference and Usage** w rozdziale Servie Control - Status.

5.3 Odczyt i manipulacja metadanymi.

Głównym zastosowaniem PROC METADATA jest jednak nie sprawdzanie stanu serwera lecz dostęp i manipulacja (tworzenie, usuwanie, modyfikacja) metadanymi. Realizowane jest to za pomogą szeregu funkcji, opisanych poniżej. Wszystkie funkcje są wywoływane według następującego schematu:

```
proc metadata
    in='<NAZWA FUKCJI>>
            <ELEMENTY SPECYFICZNE DLA FUNKCJI/>
            <NS>SAS</NS> <!-- z wyjątkiem funkcji GetRepositories-->
            <Flags>WARTOŚĆ FLAG</Flags>
            <Options>OPCJE</Options>
        </NAZWA FUNKCJI>';
run;
```

W polu <NS> ustawiana jest przestrzeń nazw dla zapytania. Poza specjalnymi przypadkami przyjmuje ona wartość "SAS".

W polu <Flags> podawana jest wartość numeryczna, reprezentująca ustawione modyfikacje domyślnego sposobu działania procedury. Wartość ta jest sumą wartości poszczególnych flag, które chcemy ustawić. Wartości poszczególnych flag, ich opis oraz funkcje w których mogą zostać one zastosowane są opisane w dokumencie **SAS® 9.4 Open Metadata Interface: Reference and Usage** w rozdziale Summary Table of IOMI Flags.

W tym samym dokumencie w rozdziale Summary Table of IOMI Options opisane zostały wartości, możliwe do zdefniowania w polu <Options>

5.3.1 Odczyt metadanych

Interfejs IOMI definiuje szereg metod pozwalających na odczyt wybranych informacji z metadanych. Są to:

5.3.1.1 GetRepositories

Funkcja ta zwraca informację o repozytoriach istniejących w metadanych. Informacja o repozytorium jest konieczna w celu wykonania innych operacji na metadanych oraz bardziej szczegółowych zapytań.

Przykład użycia:

```
proc metadata
    in='<GetRepositories>
            <Repositories/>
            <Flags>0</Flags>
            <Options/>
        </GetRepositories>';
run;
```

Wynik:

```
<GetRepositories>
    <Repositories>
        <Repository Id="A0000001.A5YA2KB5" Name="Foundation" Desc="" DefaultNS="SAS"/>
        <Repository Id="A0000001.A5RHBUM3" Name="BILineage" Desc="BILineage"
DefaultNS="SAS"/>
    </Repositories>
    <Flags>0</Flags>
    <Options/>
</GetRepositories>
```

5.3.1.2 GetTypes

Funkcja zwraca informację o wszystkich typach obiektów, zdefiniowanych w metadanych.

Przykład użycia:

```
proc metadata
    in='<GetTypes>
            <Types/>
            <NS>SAS</NS>
            <Flags>0</Flags>
            <Options/>
        </GetTypes>';
 run;
```

Wynik:

```
<GetTypes>
    <Types>
        <Type Id="AbstractExtension" Desc="Abstract extension" HasSubtypes="1"/>
        <Type Id="AbstractJob" Desc="Abstract job" HasSubtypes="1"/>
        ...
    </Types>
</GetTypes>
```

5.3.1.3 GetSubtypes

Dla większości typów zwracanych przez funkcję GetTypes istnieją zdefiniowane podtypy, opisujące specyficzne przypadki obiektów. Informacje o podtypach danego typu można uzyskać przy użyciu funkcji GetSubtypes. Przykład użycia:

```
proc metadata
    in='<GetSubtypes>
            <Supertype>DataTable</Supertype>
            <Subtypes/>
            <NS>SAS</NS>
            <Flags>0</Flags>
            <Options/>
        </GetSubtypes>';
 run;
```

Wynik:

```
<GetSubtypes>
    <Supertype>DataTable</Supertype>
    <Subtypes>
        <Type Id="ExternalTable" Desc="External data table" HasSubtypes="0"/>
        <Type Id="RelationalTable" Desc="Relational table" HasSubtypes="1"/>
        <Type Id="TableCollection" Desc="Table collection" HasSubtypes="0"/>
    </Subtypes>
    <NS>SAS</NS>
    <Flags>0</Flags>
    <Options/>
</GetSubtypes>
```

5.3.1.4 GetTypeProperties

Każdy obiekt zdefiniowany w metadanych posiada zestaw opisujących go atrybutów oraz jest powiązany z innymi obiektami. Powiązania te nazywane są asocjacjami i są one grupowane w zależności od typu zależności, jaką reprezentują. Informacje o atrybutach i asocjacjach dostępnych dla obiektu danego typu można uzyskać wykonując funkcję GetTypeProperties. Przykład użycia:

```
proc metadata
    in='<GetTypeProperties>
        <Type>Column</Type>
        <Properties/>
        <NS>SAS</NS>
        <Flags>1</Flags>
        <Options/>
        </GetTypeProperties>';
run;
```

Wynik:

```
<GetTypeProperties>
    <Type>Column</Type>
    <Properties>
        <Attributes>
            <BeginPosition Description="..." Type="Int" Length="" RequiredForAdd="No"
UpdateAllowed="Yes"/>
            <ChangeState Description="..." Type="String" Length="64" RequiredForAdd="No"
UpdateAllowed="Yes"/>
            <ColumnLength Description="..." Type="Int" Length="" RequiredForAdd="No"
UpdateAllowed="Yes"/>
            <ColumnName Description="..." Type="String" Length="256" RequiredForAdd="No"
UpdateAllowed="Yes"/>
             ...
        </Attributes>
```

```
        <Associations>
            <AccessControls AssociatedType="AccessControl" CardinalityMax="*" .../>
            <AnalyticColumns AssociatedType="AnalyticColumn" CardinalityMax="*" .../>
            ...
        </Associations>
    </Properties>
    <NS>SAS</NS>
    <Flags>1</Flags>
    <Options/>
</GetTypeProperties>
```

5.3.1.5 GetMetadataObjects oraz GetMetadata

Do tej pory przedstawiane przykłady opisywały możliwości podglądu struktury metadanych. Funkcja GetMetadataObjects oraz opisana w następnym podpunkcie GetMetadata umożliwiają dostęp do informacji o samych obiektach. Funkcja GetMetadataObjects pozwala na uzyskanie listy obiektów danego typu znajdujących się we wskazanym repozytorium. Domyślnie opis takiego obiektu składa się z jego nazwy oraz identyfikatora. Identyfikator obiektu można następnie użyć jako parametr zapytania GetMetadata w celu uzyskania reszty informacji.

5.3.1.5.1 Działanie domyślne

Poniżej przedstawiony został przykładowy proces odczytywania informacji z metadanych na przykładzie odczytu listy tabel zdefiniowanych dla biblioteki SAS-owej.

- Krok1 - wyszukujemy Id repozytorium

```
proc metadata
    in='<GetRepositories>
            <Repositories/>
            <Flags>0</Flags>
            <Options/>
        </GetRepositories>';
run;
```

Wynik:

```
<GetRepositories>
    <Repositories>
        <Repository Id="A0000001.A5YA2KB5" Name="Foundation" Desc="" DefaultNS="SAS"/>
        <Repository Id="A0000001.A5RHBUM3" Name="BILineage" Desc="BILineage"
DefaultNS="SAS"/>
    </Repositories>
    <Flags>0</Flags>
    <Options/>
</GetRepositories>
```

- Krok 2 - w ramach repozytorium Foundation wyszukujemy obiektów typu SASLibrary

```
 proc metadata
    in='<GetMetadataObjects>
            <Reposid>A0000001.A5YA2KB5</Reposid>
            <Type>SASLibrary</Type>
            <Objects/>
            <NS>SAS</NS>
            <Flags>0</Flags>
            <Options/>
        </GetMetadataObjects>';
run;
```

Wynik:

```
<GetMetadataObjects>
    <Reposid>A0000001.A5YA2KB5</Reposid>
    <Type>SASLibrary</Type>
    <Objects>
        <SASLibrary Id="A5YA2KB5.AZ000001" Name="SASApp - SASDATA"/>
        <SASLibrary Id="A5YA2KB5.AZ000002" Name="SASApp - wrsdist"/>
        <SASLibrary Id="A5YA2KB5.AZ000003" Name="SASApp - wrstemp"/>
        <SASLibrary Id="A5YA2KB5.AZ0000ST" Name="polska"/>
        <SASLibrary Id="A5YA2KB5.AZ0000SU" Name="ORACLE"/>
        <SASLibrary Id="A5YA2KB5.AZ0000SV" Name="samples"/>
        ...
    </Objects>
    <NS>SAS</NS>
    <Flags>0</Flags>
    <Options/>
</GetMetadataObjects>
```

- Krok 3 - pobieramy informację o tabelach dla biblioteki samples

```
proc metadata
    in='<GetMetadata>
            <Metadata>
                <SASLibrary Id="A5YA2KB5.AZ0000SV" Name="" Desc="">
                    <Tables/>
                </SASLibrary>
            </Metadata>
            <NS>SAS</NS>
            <Flags>0</Flags>
            <Options/>
        </GetMetadata>';
run;
```

Wynik:

```
<GetMetadata>
    <Metadata>
        <SASLibrary Id="A5YA2KB5.AZ0000SV" Name="samples" Desc="">
            <Tables>
                <PhysicalTable Id="A5YA2KB5.BI0000UD" Name="CARS" Desc=""/>
                <PhysicalTable Id="A5YA2KB5.BI0000UE" Name="CLASS" Desc=""/>
                <PhysicalTable Id="A5YA2KB5.BI0000UF" Name="HEART" Desc="Framingham Heart
Study"/>
                <PhysicalTable Id="A5YA2KB5.BI0000UG" Name="MONTH" Desc=""/>
                <PhysicalTable Id="A5YA2KB5.BI0000UH" Name="PRDSALE" Desc=""/>
                <PhysicalTable Id="A5YA2KB5.BI0000UJ" Name="RLP_CLASS" Desc=""/>
            </Tables>
        </SASLibrary>
    </Metadata>
    <NS>SAS</NS>
    <Flags>0</Flags>
    <Options/>
</GetMetadata>
```

5.3.1.5.2 Filtrowanie wyników

Jak widać na przedstawionym przykładzie, w przypadku zapytania GetMetadataObjects otrzymujemy informacje o wszystkich obiektach danego typu, co może utrudniać wyszukanie interesującej informacji. Aby temu zapobiec procedura pozwala na zastosowanie filtru. W tym celu należy w polu <Flags> ustawić wartość flagi OMI_XMLSELECT (128) oraz zdefiniować w polu <Options> wyrażenie opisujące filtr. Poniższy przykład ilustruje filtrowanie wyników po nazwie.

```
proc metadata
    in='<GetMetadataObjects>
            <Reposid>A0000001.A5YA2KB5</Reposid>
            <Type>SASLibrary</Type>
            <Objects/>
            <NS>SAS</NS>
            <Flags>128</Flags>
            <Options>
                <XMLSELECT search="*[@Name contains ''samples'']" />
            </Options>
        </GetMetadataObjects>';
run;
```

Wynik:

```
<GetMetadataObjects>
    <Reposid>A0000001.A5YA2KB5</Reposid>
    <Type>SASLibrary</Type>
    <Objects>
        <SASLibrary Id="A5YA2KB5.AZ0000SV" Name="samples"/>
    </Objects>
    <NS>SAS</NS>
    <Flags>128</Flags>
    <Options>
        <XMLSELECT search="*[@Name contains 'samples']"/>
    </Options>
</GetMetadataObjects>
```

Szczegółowe informacje na temat filtrowania wyników funkcji GetMetadataObjects można znaleźć w dokumencie **SAS® 9.4 Open Metadata Interface: Reference and Usage** w rozdziale Filtering a GetMetadataObjects Request.

5.3.1.5.3 Automatyczne wywołanie GetMetadata

GetMetadataObjects oprócz filtrowania wyników pozwala na automatyczne zwracanie wyników zapytania GetMetadata dla znalezionych obiektów. W tym celu należy w polu <Flags> ustawić wartość flagi OMI_GET_METADATA (256). W poniższym przykładzie dodatkowo ustawiamy flagi OMI_XMLSELECT (128) oraz OMI_ALL_SIMPLE (8), która powoduje, że zwracany XML zawiera wartości wszystkich atrybutów znalezionego obiektu. Przykładowy wynik:

```
proc metadata
    in='<GetMetadataObjects>
            <Reposid>A0000001.A5YA2KB5</Reposid>
            <Type>SASLibrary</Type>
            <Objects/>
            <NS>SAS</NS>
            <Flags>392</Flags><!-- 256 + 128 + 8-->
            <Options>
                <XMLSELECT search="SASLibrary[@Name = ''samples'']"/>
            </Options>
        </GetMetadataObjects>';
run;
```

Wynik:

```
<GetMetadataObjects>
    <Reposid>A0000001.A5YA2KB5</Reposid>
    <Type>SASLibrary</Type>
    <Objects>
        <SASLibrary Id="A5YA2KB5.AZ0000SV" Name="samples" ChangeState="" Desc=""
Engine="BASE" IsDBMSLibname="0" IsHidden="0" IsPreassigned="0" Libref="samples"
```

```
LockedBy="" MetadataCreated="28Aug2013:07:36:51" MetadataUpdated="28Aug2013:07:36:51"
PublicType="Library" UsageVersion="1000000"/>
    </Objects>
    <NS>SAS</NS>
    <Flags>392</Flags>
    <Options>
        <XMLSELECT search="SASLibrary[@Name = 'samples']"/>
    </Options>
</GetMetadataObjects>
```

5.3.1.5.4 Szablony (template)

Funkcja GetMetadataObjects nie musi zwracać jedynie obiektów typu zdefiniowanego w polu <Type>, lecz może również zwracać informacje o obiektach powiązanych. Aby to zrobić należy zdefiniować w polu <Options> szablon oraz dodać do pola <Flags> wartość flagi OMI_TEMPLATE (4). Przykład użycia:

```
proc metadata
    in='<GetMetadataObjects>
            <Reposid>A0000001.A5YA2KB5</Reposid>
            <Type>SASLibrary</Type>
            <Objects/>
            <NS>SAS</NS>
            <Flags>388</Flags><!-- 256 + 128 + 4 -->
            <Options>
                <Templates>
                    <SASLibrary>
                        <Tables/>
                    </SASLibrary>
                </Templates>
                <XMLSELECT search="SASLibrary[@Name = ''samples'']"/>
            </Options>
        </GetMetadataObjects>';
run;
```

Wynik:

```
<GetMetadataObjects>
    <Reposid>A0000001.A5YA2KB5</Reposid>
    <Type>SASLibrary</Type>
    <Objects>
        <SASLibrary Id="A5YA2KB5.AZ0000SV" Name="samples">
        <Tables>
                <PhysicalTable Id="A5YA2KB5.BI0000UD"/>
                <PhysicalTable Id="A5YA2KB5.BI0000UE"/>
                <PhysicalTable Id="A5YA2KB5.BI0000UF"/>
                <PhysicalTable Id="A5YA2KB5.BI0000UG"/>
                <PhysicalTable Id="A5YA2KB5.BI0000UH"/>
                <PhysicalTable Id="A5YA2KB5.BI0000UJ"/>
            </Tables>
        </SASLibrary>
    </Objects>
    <NS>SAS</NS>
    <Flags>388</Flags>
    <Options>
        <Templates>
            <SASLibrary>
                <Tables/>
            </SASLibrary>
        </Templates>
        <XMLSELECT search="SASLibrary[@Name = 'samples']"/>
    </Options>
</GetMetadataObjects>
```

Szczegółowe informacje na temat stosowania szablonów w wywołaniach funkcji GetMetadataObjects można znaleźć w dokumencie **SAS® 9.4 Open Metadata Interface: Reference and Usage** w rozdziale Expanding a GetMetadataObjects Request to Return Additional Properties.

5.3.2 Dodawanie metadanych - AddMetadata

Do tworzenia obiektów w metadanych służy funkcja AddMetadata. Pozwala ona na tworzenie nowego obiektu wraz z asocjacjami do innych obiektów. Do utworzenia nowego obiektu konieczna jest znajomość Id repozytorium oraz definicja obiektu zapisana w postaci XML w polu metadata. Poniżej znajduje się przykładowe wywołanie funkcji AddMetadata, tworzące nowy obiekt tabeli oraz przypisujące ten obiekt do biblioteki ***samples*** z poprzednich przykładów.

```
proc metadata
   in='<AddMetadata>
          <Metadata>
              <PhysicalTable Name="dummy_table"
              Desc="Dummy Table">
                  <TablePackage>
                      <SASLibrary ObjRef="A5YA2KB5.AZ0000SV"/><!--id biblioteki samples-->
                  </TablePackage>
              </PhysicalTable>
          </Metadata>
          <Reposid>A0000001.A5YA2KB5</Reposid>
          <NS>SAS</NS>
          <Flags>268435456</Flags> <!-- OMI_TRUSTED_CLIENT, wartość wymagana -->
          <Options/>
      </AddMetadata>';
run;
```

Po wykonaniu powyższego kodu widzimy w SAS Management Console, że obiekt został utworzony i poprawnie przypisany do biblioteki ***samples***. Świadczy o tym poniższy zrzut ekranu.

Name	Description	Location
CARS		/Shared Data/02 Tables/SAMPLES
CLASS		/Shared Data/02 Tables/SAMPLES
dummy_table	Dummy Table	
HEART	Framingham Heart Study	/Shared Data/02 Tables/SAMPLES
MONTH		/Shared Data/02 Tables/SAMPLES
PRDSALE		/Shared Data/02 Tables/SAMPLES
RLP_CLASS		/Shared Data

Widzimy również, iż utworzony obiekt nie zawiera wszystkich niezbędnych wartości, takich jak lokalizacja. Ich uwzględnienie w tworzonym obiekcie wymagałoby odpowiedniej rozbudowy opisu obiektu w sekcji <Metadata>.

Pełny opis użycia funkcji AddMetadata znajduje się w dokumencie **SAS® 9.4 Open Metadata Interface: Reference and Usage** w rozdziale Adding Metadata Objects.

5.3.3 Aktualizacja metadanych – UpdateMetadata

Funkcja AddMetadata opisywana w punkcie 5.4 pozwala jedynie na tworzenie nowych obiektów. Jeżeli chcemy dokonać modyfikacji istniejącego obiektu, czy to poprzez modyfikację jego atrybutów czy zmianę asocjacji do innych obiektów, konieczne jest użycie funkcji UpdateMetadata.

W celu modyfikacji wartości atrybutów obiektu musimy wskazać go poprzez Id oraz podać nowe wartości dla zmienianych wartości. W przypadku modyfikacji asocjacji, konieczne jest zdefiniowanie typu modyfikacji. Dostępne są następujące opcje modyfikacji asocjacji:

- Append – dodanie nowego połączenia do obiektu dla danego typu asocjacji
- Merge – dodanie lub modyfikacja połączenia dla danego typu asocjacji
- Modify – modyfikuje istniejące połączenie używając przesłanych definicji. Jeżeli asocjacja wskazana do modyfikacji nie istnieje, jest ona tworzona.
- Replace – nadpisuje całą listę asocjacji listą zdefiniowaną w poleceniu
- Remove – usuwa asocjację wskazaną w poleceniu bez modyfikacji pozostałych połączeń

Przykład użycia funkcji UpdateMetadana przedstawiony został poniżej.

```
proc metadata
    in='<UpdateMetadata>
            <Metadata>
                <PhysicalTable Id="A5YA2KB5.BI0000UL">
                    <TablePackage Function="Remove">
                        <!-- Id obiektu biblioteki samples -->
                        <SASLibrary Id="A5YA2KB5.AZ0000SV"/>
                    </TablePackage>
                </PhysicalTable>
            </Metadata>
            <NS>SAS</NS>
            <Flags>268435456</Flags> <!-- OMI_TRUSTED_CLIENT -->
            <Options/>
        </UpdateMetadata>';
run;
```

W wyniku wykonania powyższego kodu usunięte zastało powiązanie pomiędzy obiektem tabeli ***Dummy Table*** a biblioteką ***samples***. W rezultacie obiekt tabeli w dalszym ciągu istnieje, lecz nie jest on dostępny z poziomu biblioteki, co pokazuje poniższy zrzut ekranu.

Name	Description	Location
CARS		/Shared Data/02 Tables/SAMPLES
CLASS		/Shared Data/02 Tables/SAMPLES
HEART	Framingham Heart Study	/Shared Data/02 Tables/SAMPLES
MONTH		/Shared Data/02 Tables/SAMPLES
PRDSALE		/Shared Data/02 Tables/SAMPLES
RLP_CLASS		/Shared Data

Pełny opis użycia funkcji UpdateMetadata znajduje się w dokumencie **SAS® 9.4 Open Metadata Interface: Reference and Usage** w rozdziale Updating Metadata Objects.

5.3.4 Usuwanie metadanych – DeleteMetadata

W wyniku wykonania kodu z przykładu w punkcie 5.5 otrzymaliśmy obiekt tabeli, nie podłączony do żadnej biblioteki. Obiekt taki nie jest dostępny dla użytkownika w jakiejkolwiek aplikacji klienckiej, co czyni go bezużytecznym. Aby usunąć taki obiekt z metadanych, należy wykonać funkcję DeleteMetadata. Poniżej zamieszczone zostało przykładowe wywołanie tej funkcji, usuwające obiekt tabeli ***Dummy Table***. W celu wyświetlenia listy usuniętych obiektów w polu <Flags> oprócz obowiązkowej flagi OMI_TRUSTED_CLIENT została dodatkowo ustawiona flaga OMI_RETURN_LIST (1024).

```
proc metadata
    in='<DeleteMetadata>
            <Metadata>
                <PhysicalTable Id="A5YA2KB5.BI0000UL"/>
            </Metadata>
        <NS>SAS</NS>
        <Flags>268436480</Flags> <!-- OMI_RETURN_LIST + OMI_TRUSTED_CLIENT -->
        <Options/>
        </DeleteMetadata>';
run;
```

Wynik:

```
<DeleteMetadata>
    <Metadata>
        <PhysicalTable Id="A5YA2KB5.BI0000UL"/>
    </Metadata>
    <NS>SAS</NS>
    <Flags>268436480</Flags>
    <Options/>
</DeleteMetadata>
```

W powyższym przykładzie ustawione zostały następujące flagi: OMI_TRUSTED_CLIENT, obowiązkowa przy wszystkich modyfikacjach metadanych oraz OMI_RETURN_LIST, dzięki której w wyniku wykonania procedury otrzymujemy informację, które obiekty zostały usunięte.

Potwierdzeniem, że obiekt został faktycznie usunięty może być również ponowne uruchomienie operacji usunięcia. Jeżeli obiekt, który ma zostać usunięty, nie istnieje, procedura zwraca następujący błąd:

```
ERROR: PhysicalTable : A5YA2KB5.BI0000UL cannot be found in the ptable container in the
Foundation repository.
```

Pełny opis użycia funkcji DeleteMetadata znajduje się w dokumencie **SAS® 9.4 Open Metadata Interface: Reference and Usage** w rozdziale Deleting Metadata Objects.

6 Dostęp do metadanych z poziomu DATA Step-u.

6.1 Dostępne funkcje

Funkcje opisane w punkcie 5 są pozwalają na pełną manipulację metadanymi. Są one jednak pomyślane do wykorzystania przez twórców aplikacji klienckich bądź zewnętrznych narzędzi, współpracujących z serwerem metadanych. Fakt, że wysyłane zapytania oraz otrzymywane odpowiedzi są przekazywane w postaci dokumentów XML czyni je trudnymi do zastosowania w programach napisanych w SAS 4GL. W tym przypadku, rozwiązaniem jest użycie funkcji dostępu do metadanych z poziomu DATA-stepu. Funkcje te to:

- METADATA_APPPROP – zwraca wartość wybranego parametru obiektu SoftwareComponent bądź DeployedComponent
- METADATA_DELASSN – usuwa wszystkie obiekty tworzące wskazaną asocjację
- METADATA_DELOBJ – usuwa pierwszy obiekt pasujący do wskazanego URI
- METADATA_GETATTR – zwraca wartość wskazanego atrybutu dla wybranego obiektu
- METADATA_GETNASL – zwraca n-tą asocjację danego obiektu
- METADATA_GETNASN – zwraca n-ty obiekt tworzący wskazaną asocjację
- METADATA_GETNATTR – zwraca n-ty atrybut obiektu o wskazanym URI
- METADATA_GETNOBJ – zwraca n-ty obiekt pasujący do wskazanego URI
- METADATA_GETNPRP – zwraca n-ty parametr wskazanego obiektu
- METADATA_GETNTYP – zwraca n-ty typ obiektu z metadanych
- METADATA_GETPROP – zwraca wartość wskazanego parametru dla wskazanego obiektu
- METADATA_NEWOBJ – tworzy nowy obiekt w metadanych
- METADATA_PATHOBJ – zwraca Id oraz typ dla wskazanego obiektu folderu
- METADATA_PAUSED – sprawdza czy serwer metadanych jest zatrzymany
- METADATA_PURGE – usuwa wskazany obiekt z pamięci cache serwera.
- METADATA_RESOLVE – zwraca identyfikator oraz typ obiektu odpowiadającego wskazanemu URI
- METADATA_SETASSN – modyfikuje wskazaną listę asocjacji dla wskazanego obiektu
- METADATA_SETATTR – modyfikuje wartość wskazanego atrybutu dla wskazanego obiektu
- METADATA_SETPROP – modyfikuje wartość wskazanego atrybutu rozszerzonego dla wskazanego obiektu
- METADATA_VERSION – zwraca wersję modelu metadanych wykorzystywanego przez serwer.

Szczegółowe informacje na temat poszczególnych funkcji znajdują się w dokumencie **SAS®9.4 Language Interfaces to Metadata** w rozdziale DATA Step Functions for Reading and Writing Metadata.

6.2 Przykłady

6.2.1 Dodawanie użytkownika do grupy.

Poniższy kod obrazuje proces dodawania użytkownika do grupy. Operacja ta jest realizowana poprzez dodanie odpowiedniej asocjacji między obiektem użytkownika a obiektem grupy.

```
%macro add_user_to_group(user, group);
    data _null_;
        length usr_uri grp_uri $256;

        /* Wyszukanie obiektu reprezentującego użytkownika o wskazanej nazwie */
        nr = 1;
        rc = metadata_getnobj("omsobj:Person?@Name contains '&user'", nr, usr_uri);
        put rc= usr_uri =;
        if (rc eq -4) then do;
            put "Użytkownik &user nie istnieje.";
            stop;
        end;

        /* Wyszukanie obiektu reprezentującego grupę o wskazanej nazwie */
        nr =1 ;
        rc = metadata_getnobj("omsobj:IdentityGroup?@Name contains '&group'", nr,
grp_uri);
        put rc= grp_uri =;
        if (rc eq -4) then do;
            put "Grupa &group nie istnieje.";
            stop;
        end;

        /* Dodanie asocjacji typu IdentityGroups między obiektem użytkownika a obiektem
grupy */
        rc = metadata_setassn(usr_uri, "IdentityGroups", "Append", grp_uri);
        if (rc ne 0) then do;
            put "Błąd podczas dodawania uzytkownika do grupy.";
            stop;
        end;
    run;
%mend;

%add_user_to_group(sasdemo, Group_B);
```

6.2.2 Rejestracja biblioteki SAS BASE w metadanych.

Poniższy kod przedstawia proces rejestracji biblioteki SAS-owej w metadanych. Razem z biblioteką rejestrowane są wszystkie znajdujące się w niej tabele. Dla każdej z tabel rejestrowane są informacje o kolumnach. Dla uproszczenia przykładu wszystkie obiekty są zapisywane w katalogu ***Shared Data***.

Krok 1 – rejestracja biblioteki

```
%macro register_sas_library(name, libref, path);
    data _null_;
        length lib_uri dir_uri ctx_uri   $256;
        rc=0;

        /* Tworzymy obiekt biblioteki (SASLibrary) w folderze Shared Data */
        rc=metadata_newobj("SASLibrary", lib_uri, &name, "Foundation",
"omsobj:Tree?@Name='Shared Data'", "Members");
        put rc=;
        put lib_uri=;
        /* Ustawiamy wartość atrybutów PublicType and UsageVersion */
        rc=metadata_setattr(lib_uri, "PublicType","Library");
        put rc=;

        rc=metadata_setattr(lib_uri, "UsageVersion", "1000000.0");
        put rc=;
        /* Ustawiamy wartość atrybutów Libref and Engine */
        rc=metadata_setattr(lib_uri, "Libref", &libref);
        put rc=;
        rc=metadata_setattr(lib_uri, "Engine", "BASE");
        put rc=;

        /* Tworzymy obiekt katalogu biblioteki */
        rc=metadata_newobj("Directory", dir_uri, "Path", "Foundation", lib_uri,
"UsingPackages");
        put rc=;
        put dir_uri=;
        /* Ustawiamy wartość atrybutówDirectoryRole and DirectoryName */
        rc=metadata_setattr(dir_uri, "DirectoryRole", "LibraryPath");
        put rc=;
        rc=metadata_setattr(dir_uri, "DirectoryName", &path);
        put rc=;
        rc=metadata_setattr(dir_uri, "UsageVersion", "1000000");
        put rc=;
        /* Ustawiamy że bibliotekę będzie dostępna na serwerze SASApp*/
        rc=metadata_getnobj("omsobj:ServerContext?@Name contains 'SASApp'", 1, ctx_uri);
        put rc=;
        put ctx_uri=;

        rc = metadata_setassn(lib_uri, "DeployedComponents", "Append", ctx_uri);
        put rc=;

        call execute('%register_tables(' || lib_uri || ', ' || &path || ')');
    run;
%mend;
```

Krok 2 – rejestracja tabel z biblioteki

```
%macro register_tables(lib_uri, lib_path);
    libname temp "&lib_path";
    /* Pobieramy informacje o tabelach w bibliotece */
    proc sql;
        create table temp_tables as
        select memname as name from sashelp.vtable
        where libname = upcase('temp');
    quit;

    data _null_;
        length tbl_uri $256;
        set temp_tables;
        /* Tworzymy obiekt tabeli w katalogu Shared Data */
        rc = metadata_newobj("PhysicalTable", tbl_uri, name, "Foundation",
"omsobj:Tree?@Name='Shared Data'", "Members");
        put rc=;
        put tbl_uri=;
        /* Ustawiamy dodatkowe atrybuty obiektu tabeli*/
        rc=metadata_setattr(tbl_uri, "PublicType", "Table");
        put rc=;
        rc=metadata_setattr(tbl_uri, "UsageVersion", "1000000");
        put rc=;
        rc=metadata_setattr(tbl_uri, "MemberType", "DATA");
        put rc=;
        rc=metadata_setattr(tbl_uri, "TableName", upcase(name));
        put rc=;
        rc=metadata_setattr(tbl_uri, "SASTableName", upcase(name));
        /* Dodajemy powiązanie między biblioteką a tabelą */
        rc=metadata_setassn("&lib_uri", "Tables", "Append", tbl_uri);
        put rc=;
        put "&lib_uri";

        call execute('%register_table_columns(' || tbl_uri || ', ' || name || ')');
    run;

    proc datasets library=work noprint;
        delete temp_tables;
    run;

    libname temp clear;

%mend;
```

Krok 3 – rejestracja kolumn

```
%macro register_table_columns(tbl_uri, tbl_name);
    /* Pobieramy informacje o kolumnach w danej tabeli */
    proc sql;
        create table temp_cols as
        select * from sashelp.vcolumn
        where libname = upcase('temp') and memname = upcase("&tbl_name");
    quit;

    data _null_;
        set temp_cols;
        length col_uri $256;
        /* Tworzymy obiekt columny powiązany w aktualną tabelą */
        rc=metadata_newobj("Column", col_uri, name, "Foundation", "&tbl_uri", "Columns");
        put col_uri=;
        put rc=;
        /* Ustawiamy parametry obiektu. */
        rc=metadata_setattr(col_uri, "PublicType", "Column");
        put rc=;
        rc=metadata_setattr(col_uri, "UsageVersion", "1000000");
        put rc=;
        /* Zapisujemy parametry kolumny w metadanych */
        rc=metadata_setattr(col_uri, "ColumnLength", length);
        put rc=;
        rc=metadata_setattr(col_uri, "SASColumnLength", length);
        put rc=;
        rc=metadata_setattr(col_uri, "ColumnName", name);
        put rc=;
        rc=metadata_setattr(col_uri, "SASColumnName", name);
        put rc=;
        if (type eq 'char') then do;
            rc=metadata_setattr(col_uri, "SASColumnType", 'C');
            put rc=;
        end;
        else do;
            rc=metadata_setattr(col_uri, "SASColumnType", 'N');
            put rc=;
        end;
        rc=metadata_setattr(col_uri, "SASFormat", format);
        put rc=;
    run;


    proc datasets library=work noprint;
        delete temp_cols;
    run;
%mend;
```

Przykładowe wywołanie - %***register_sas_library***("DS._TEST", "ds_test", "C:\DS_TEST");