

PLADENJ B51 - INTEROPERABILNOSTNI SISTEM ZA IZVAJANJE ELEKTRONSKIH POIZVEDB

Tip dokumenta	SPECIFIKACIJA
Avtor	MAG. PETAR BRAJAK
Datum zadnje verzije	13. FEBRUAR, 2017

Kazalo

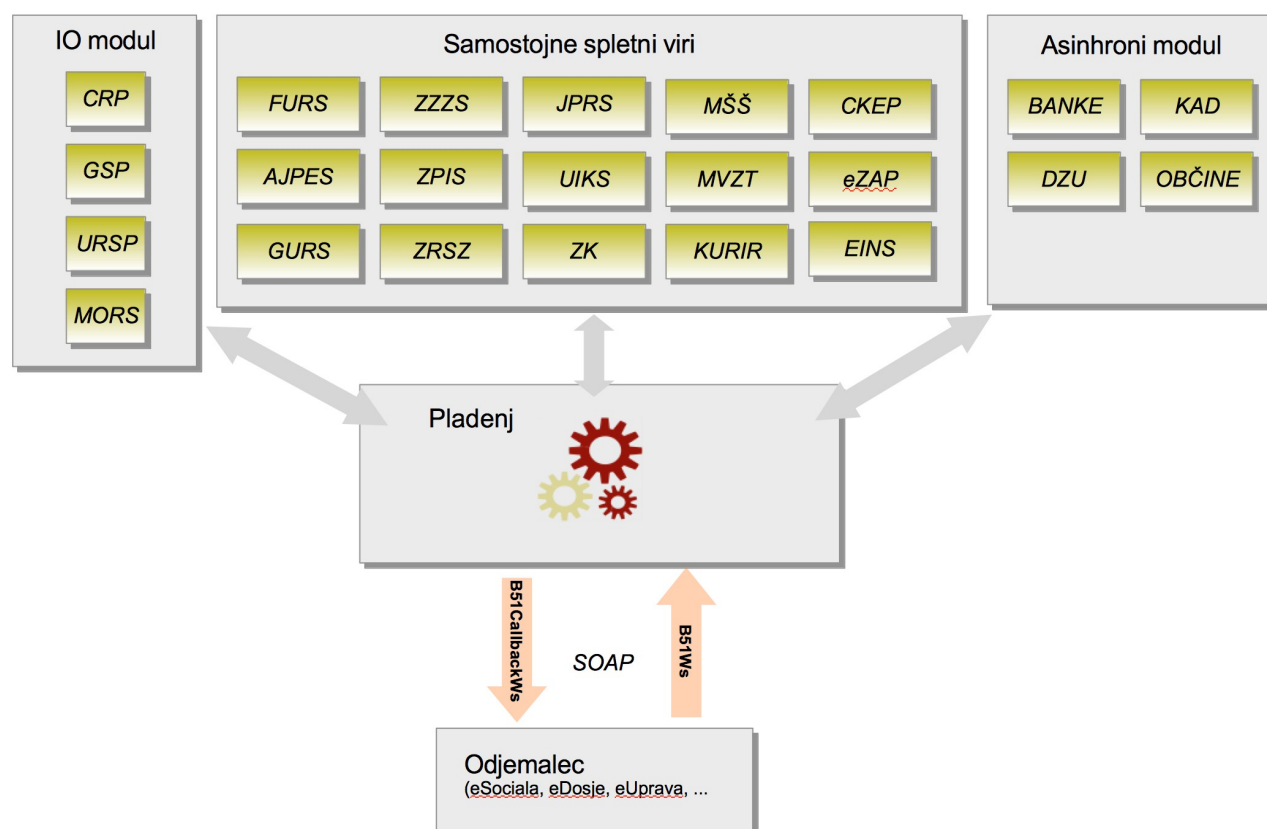
1 UVOD	3
2 ZNAČILNOSTI SISTEMA PLADENJ	4
3 PROTOKOL PLADENJ – ODJEMALEC	6
3.1 <i>Zahtevke – spletna storitev B51Ws</i>	7
3.1.1 Metoda handleRequest	8
3.1.1.1 Sinhroni klic – metoda handleRequest3	11
3.1.2 Postopek	12
3.2 <i>Pridobivanje podatkov – spletna storitev B51Ws</i>	15
3.2.1 Dekriptiranje pridobljenih podatkov	18
3.3 <i>Obveščanje – spletna storitev B51CallackWs</i>	20
3.4 <i>Zaključek postopka – spletna storitev B51Ws</i>	22
4 VKLJUČEVANJE NOVEGA ODJEMALCA NA PLADENJ	23

1 UVOD

B51-Pladenj je centralni, horizontalni informacijski sistem za izvajanje elektronskih poizvedb, ki na osnovi najsodobnejših konceptov storitveno usmerjene arhitekture (SOA) omogoča dinamično gradnjo postopkov za pridobivanje logično povezanih podatkov iz različnih podatkovnih virov v realnem času. Sistem omogoča tudi varno in zanesljivo hranjenje pridobljenih podatkov bodisi v transakcijski ali arhivski zbirki. Zahtevki za pridobivanje podatkov se v realnem času preslikajo v BPM procese s katerimi Pladenj enoznačno določi podatkovne vire in pripadajoče poizvedbe in uporabnikom sistema „Pladenj“ (aplikacija odjemalca) pridobi vse pričakovane podatke.

Osnovna ideja sistema je v tem, da se aplikaciji odjemalca čim bolj poenostavi tehnična kompleksnost pridobivanja podatkov iz različnih podatkovnih virov in na ta način omogoči aplikaciji osredotočenost na interpretacijo vsebine podatkov in ne na proces pridobivanja teh podatkov. To zmanjšanje tehnične kompleksnosti pridobivanja podatkov je doseženo z vzpostavitvijo centralne spletne točke, ki sprejema podatkovne zahteve od odjemalcev in jih po določenih pravilih transformira v množico atomarnih poizvedb na konkretne podatkovne vire, zbere odgovore od teh podatkovnih virov in jih na zanesljiv način dostavi odjemalcu za nadaljnjo obdelavo.

Dokument opisuje spletni protokol komunikacije med sistemom Pladenj in odjemalci. Namenjen je razvijalcem aplikacij, ki potrebuje podatke iz podatkovnih virov. Protokol je določen s spletnimi storitvami B51Ws in B51CallbackWs, ki popolnoma skrijeta tehnološko in vsebinsko kompleksnost sistema, ter zagotovita visok nivo abstrakcije pridobivanja podatkov iz tehnološko in vsebinsko različnih podatkovnih virov. Naslednja slika prikazuje vlogo sistema Pladenj in odnos do podatkovnih virov in odjemalcev:



Slika 1 – Vloga sistema Pladenj

Protokol komunikacije vključuje naslednje korake:

1. Odjemalec sproži zahtevek za začetek pridobivanja podatkov preko metode `handleRequest` spletne storitve `B51Ws`. Parameter metode je XML objekt po shemi `B51RequestType`, ki določi akcijo ter pogoje za pridobivanje podatkov: namen, emšo, davčna, referenčni datumi, itd.
2. Pladenj izvede pridobivanje podatkov od različnih podatkovnih virov in shrani zakodirane odgovore teh virov v začasno, transakcijsko zbirko. Pladenj sproti obvešča odjemalca o statusu pridobljenih podatkov preko metode `handleResponse`. Parameter spletne storitve `B51CallbackWs` je XML objekt po shemi `B51ResponseType`, ki vsebuje meta podatke in status vseh podatkovnih virov vključenih v poizvedovanje.
3. Odjemalec na lastno iniciativo prinese podatke (XML objekt z delnimi ali vsemi pridobljenimi podatki) preko metode `getTransactionData` spletne storitve `B51Ws`. Odjemalec odkodira XML, ga razčleni in uporabi za svoje potrebe. Odjemalec lahko večkrat kliče metodo `getTransactionData`. Odgovori od podatkovnih virov so zaviti v XML ovojnico skupaj z meta podatki po XSD shemi `B51ResponseType`.
4. Odjemalec preko metode `handleRequest` spletne storitve `B51Ws` obvesti Pladenj, da ne potrebuje več podatkov. Pladenj izvede brisanje vsebine podatkov, ter arhiviranje meta podatkov za potrebe revizijske sledi.

2 ZNAČILNOSTI SISTEMA PLADENJ

Najbolj pomembna lastnost sistema Pladenj je njegova izredna fleksibilnost. Sistem zagotavlja administracijo postopkov s katerim se opredeli način pridobivanja podatkov iz različnih virov. Administrativni modul omogoča enostavno dodajanje novih podatkovnih virov in njihovo vključevanje v različne sestavljene postopke. Omogoča popolno parametrizacijo klicev spletnih metod virov preko XML predlog ter določa različne načine izvajanja poizvedb s katerimi se zagotovi večja odzivnost in prepustnost sistema. Administracija omogoča tudi gradnjo medsebojno odvisnih poizvedb (npr. izhodni podatki ene poizvedbe so lahko vhodni podatki druge poizvedbe, itd.) in posledično gradnjo zelo kompleksnih postopkov pridobivanja podatkov. Ključne vsebinske lastnosti sistema Pladenj so naslednje:

1. Pladenj zagotavlja visok nivo abstrakcije dostopa do podatkov podatkovnih virov. Odjemalec se ne ukvarja s tehnološkimi in sistemskimi značilnostmi vsakega posameznega vira (lokacija, način integracije, transportni protokol, združljivost programske in strojne opreme, varnost, itd.), ampak se koncentrira na interpretacijo vsebin pridobljenih podatkov, ter krmiljenje zahtevkov oz. postopkov. Pladenj je zgrajen na konceptu vzorca ločitve skrbi (*ang. Pattern Separation of Concern*) in njegova skrb je pridobivanje podatkov na enoten, zanesljiv, varen in hiter način. Skrb odjemalcev je poslovna logika in algoritmi odločanja na osnovi pridobljenih podatkov. Skrb podatkovnih virov je priprava kvalitetnih podatkov v obliki in vsebini kot jo potrebujejo odjemalci.
2. Pladenj zagotavlja enotno vhodno točko na osnovi SOAP spletne storitve in enotno vhodno podatkovno XSD shemo `B51RequestType` za vse odjemalce. Odjemalci ne potrebujejo neposrednih logičnih in fizičnih povezav do N različnih podatkovnih virov. Pladenj je v vlogi inteligentnega storitvenega vodila, prilagojenega in optimiziranega za potrebe pridobivanja podatkov iz več podatkovnih virov hkrati.
3. Pladenj zagotavlja enoten format odgovora po XSD shemi `B51ResponseType`.

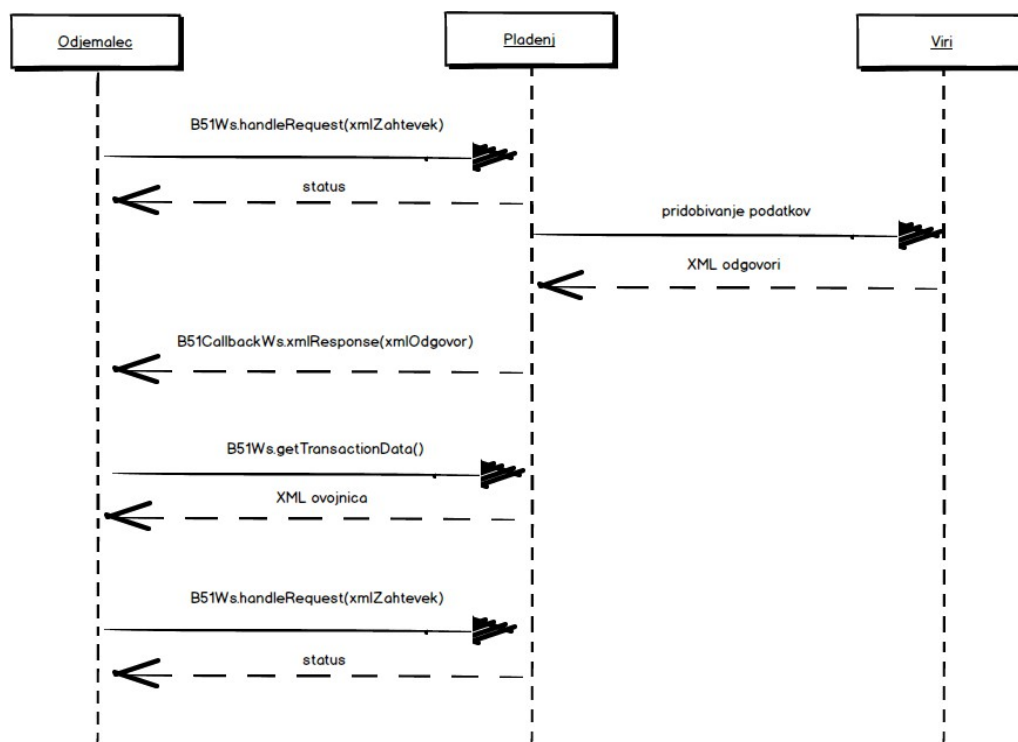
4. Pladenj omogoča logično kombiniranje pridobivanja podatkov od virov v postopku. Postopki se določijo preko administrativnih mask. Izvajanje postopkov se v realnem času preslika v dinamične BPM procese. Omogočeno je tudi kombiniranje postopkov tako, da Pladenj naredi unijo poizvedb iz različnih postopkov.
5. Pladenj zagotavlja pravilnost izvajanja vrstnega reda pridobivanja podatkov. Omogoča gradnjo povezanih podatkovnih virov tako, da so odgovori enega podatkovnega vira vhodni parametri za drugi podatkovni vir.
6. V postopku je mogoče določiti vzporedno pridobivanje podatkov iz več virov. S tem se bistveno pohitri način pridobivanja podatkov iz počasnih virov.
7. Za vsak podatkovni vir je možno določiti hitrost poizvedovanja oz. število hkratnih poizvedb. Na ta način je možno izvajati fino nastavitve in uglaševanje prepustnosti sistema in virov.
8. Pladenj podpira 4 različne načine komunikacije z viri:
 - sinhron – ob klicu se čaka na odgovor vira,
 - asinhron – pošlje se zahtevek, spletni klic od vira sproži nadaljevanje procesa,
 - paketni – zbere se več zahtevkov v paket in sproži asinhroni način komunikacije,
 - pooling – pošlje se zahtevek, ter se sproti preverja status odgovora od vira.
9. Pladenj zagotavlja odjemalcu, da so pridobljeni podatki preverjeni po XSD shemi vsakega podatkovnega vira. Odjemalec poleg prejetega XML odgovora dobi tudi informacijo o skladnosti s pripadajočo XSD shemo.
10. Vsi odgovori so kodirani s ključem odjemalca tako, da so podatki popolnoma zaščiteni pred branjem nepooblaščenih odjemalcev.
11. Revizijska sled se vodi za vsako poizvedbo. Meta podatki odgovorov so shranjeni v arhivski zbirki.
12. Ker je realno pričakovati, da pri delovanju podatkovnega vira lahko nastopijo tudi občasne motnje, Pladenj vključuje tudi samo-ponovitveno funkcionalnost (t.i. auto-resume), ki omogoča večkratno ponavljanje poizvedb na podatkovni vir v vnaprej določenih intervalih in posredovanje odgovora uporabniku potem, ko se poizvedba uspešno izvede.
13. Omogočen je tudi scenarij ponovnega zagona postopka (akcija RESUME) v primeru, da postopek konča z napako, kot tudi preklica postopka (akcija CANCEL) v primeru, da se odjemalec odloči predčasno končati postopek. Osnovna scenarija uporabe sta akciji START in STOP oz. zahtevek za pridobivanje podatkov, ter zaključevanje transakcije.
14. Proces izvajanja postopka je zgrajen na osnovi dinamičnega združevanja BPM mikro procesov. Vsak mikro proces predstavlja specifično funkcionalnost za pridobivanje podatkov iz določenega podatkovnega vira.
15. Pladenj zagotavlja administrativne maske za urejanje informacij, ki so potrebne za popolno parametrizacijo podatkov o institucijah, virih, poizvedbah, postopkih in varnostni shemi.
16. Pladenj vključuje grafični vmesnik za iskanje meta podatkov poizvedb po arhivski zbirki podatkov.
17. Vgrajena je tudi posebna funkcionalnost Monitor, ki omogoča stalno spremljanje dosegljivosti podatkovnih virov in takojšnje obveščanja in ukrepanje v primeru izpada nekega podatkovnega vira.

Tehnološko, Pladenj uporablja najsodobnejšo standardno, odprtokodno javansko SOA tehnologijo kar pomeni, da uporablja SOAP, REST, HTTPS protokol in XML/XSD sheme za zagotavljanje povezljivosti z različnimi podatkovnimi viri, koncept BPM (Business Process Management) za orkestracijo velikega števila postopkov v realnem času, itd. Pri razvoju so bile uporabljene najsodobnejše tehnike programiranja strežniških javanskih komponent (EJB 3 entitete, sejna zrna, JMS, JAAS varnost, OR preslikave, itd.) in uporabljeni uveljavljeni integracijski standardi (XML, SOAP, JAX-WS) s fokusom na paralelizem sistema, veliko prepustnosti in odzivnost. Sistem zagotavlja linearno rast in visoko razpoložljivost, ter razbremenitev podatkovne baze in ga je možno enostavno vgraditi v „oblak“.

Pladenj je zasnovan na konceptu dinamične gradnje poslovnih procesov v okolju Java Enterprise Edition aplikacijskega strežnika. Jedro rešitve je inovativna BPM arhitektura, ki je sposobna iz manjših vnaprej pripravljenih gradnikov (mikro procesov), dinamično in v realnem času zgraditi BPM proces za katerokoli smiselno kombinacijo poizvedb do podatkovnih virov. To pomeni, da je bistvena in ključna lastnost glavnega oz. osnovnega procesa zmožnost, da v posameznih točkah procesa dinamično doda podrejene procese (mikro procese), ki so specifični za določeni vir ali način pridobivanja podatkov. Na ta način se število scenarijev uporabe v sistemu bistveno zmanjša in je vzdrževanje sistema lažje in cenejše.

3 PROTOKOL PLADENJ – ODJEMALEC

Protokol komunikacije med sistemom Pladenj in odjemalci je zagotovljen preko spletne storitve B51Ws, ki jo implementira Pladenj in spletne storitve B51CallackWs, ki jo implementira odjemalec. Na naslednjem diagramu vidimo zaporedje klicanja metod:



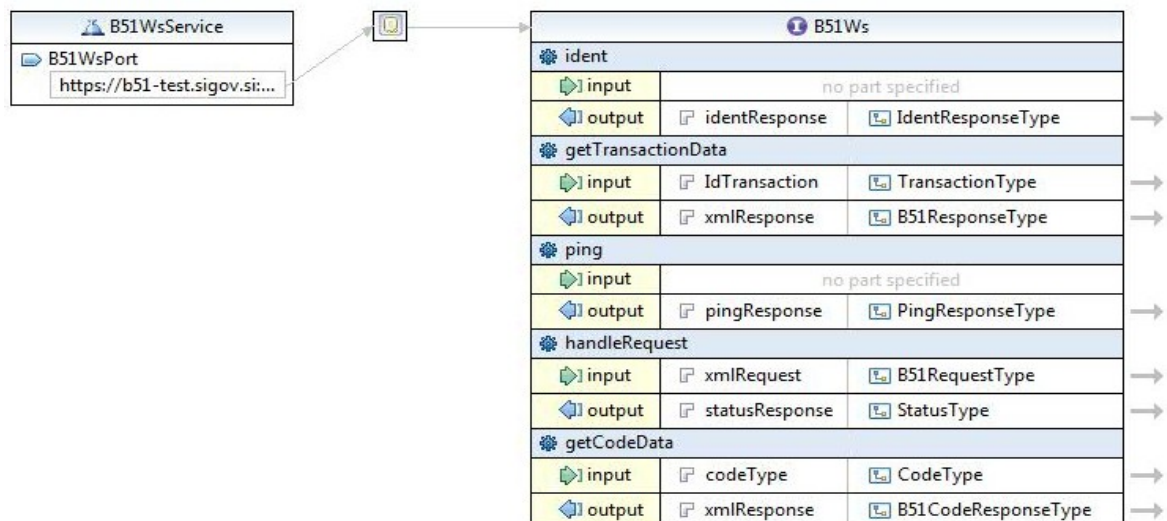
Slika 2 – Protokol klicanja

1. Odjemalec kliče metodo `handleRequest` spletne storitve `B51Ws`. Vhodni XML je tipa `B51RequestType` z akcijo `START` in parametri kot so davčna številka, emšo, referenčni datumi, itd.

2. Pladenj preveri vhodne parameter in avtorizacijo ter vrne status o uspehu preverjanja vhodnih parametrov.
3. Pladenj začne BPM proces pridobivanja podatkov od različnih podatkovnih virov. Odgovore shrani zakodirane v transakcijsko bazo.
4. Pladenj obvesti odjemalca o statusu pridobljenih podatkov preko spletne storitve `B51CallbackWs`. Vhodni parameter je XML tipa `B51ResponseType`. XML vsebuje status uspešnosti pridobivanja podatkov za vsak podatkovni vir.
5. Odjemalec vrne status uspešnega prejema spletnega klica.
6. Odjemalec kliče metodo `getTransactionData` spletne storitve `B51Ws`. Odgovor metode je XML tipa `B51ResponseType`. XML je ovojnica podatkov iz podatkovnih virov in meta podatkov.
7. Odjemalec kliče metodo `handleRequest` spletne storitve `B51Ws`. Vhodni XML je tipa `B51RequestType` z akcijo STOP. Pladenj shrani meta podatke v arhivsko zbirko in zbriše vse podatke iz transakcijske zbirke.

3.1 Zahtevek – spletna storitev B51Ws

Spletna storitev `B51Ws` je vhodna točka v sistem Pladenj. Spletna storitev je opisana z datoteko `B51Ws.wsdl`. Dostop do metod spletne storitve je omogočen uporabnikom kvalificiranih digitalnih potrdil, ki so v sistemu Pladenj administrirani v posebni vlogi za izvajanje postopkov pridobivanja podatkov iz podatkovnih virov. Torej, za uspešen klic metod spletnega servisa `B51Ws` je potrebno pridobiti in administrirati certifikat v varnostno shemo sistema Pladenj. Identifikacija in avtorizacija je implementirana s pomočjo storitve JAAS (Java Authentication and Authorisation Service) tako, da aplikacija avtorizira semantično pravilnost klica in vhodnih parametrov za prijavljenega uporabnika (npr. katere postopke uporabnik lahko pokliče). Imenski prostor je <http://nio.gov.si/names/pladenj/entrypoint>.

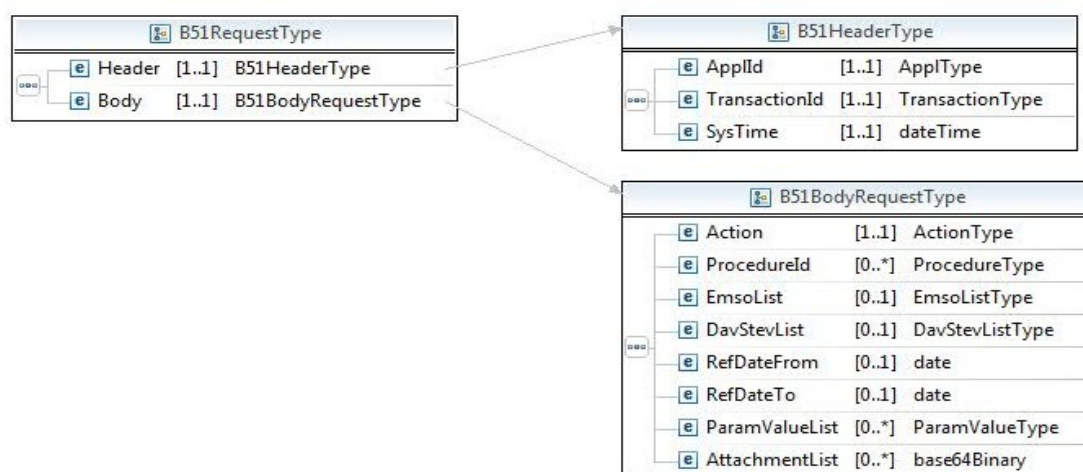


Slika 3 - Spletna storitev B51Ws

3.1.1 Metoda `handleRequest`

Metoda `handleRequest` je ključna za začetek izvajanja procesa pridobivanja podatkov preko sistema Pladenj. To je sinhrona metoda, ki takoj vrne odgovor odjemalcu, bodisi OK v primeru pravilnih vhodnih parametrov, ali ERROR v primeru neveljavnih vhodnih parametrov. V primeru veljavnih parametrov, metoda `handleRequest` sproži asinhroni postopek pridobivanja podatkov iz podatkovnih virov. Zahtevek se pošlje v čakalno vrsto, ki zagotavlja optimalno prepustnost sistema glede na trenutno obremenitev. Na osnovi vhodne informacije o šifri postopka, Pladenj začne orkestracijo več BPM procesov. Število procesov je odvisno od števila poizvedb do podatkovnih virov, načinu pridobivanja podatkov (vzporedno/zaporedno), prepustnosti, itd.

Vhodni parameter metode je XML objekt strukture `B51RequestType`:



Slika 4 - Vhodna XSD struktura

Glava XML objekta (`B51HeaderType`) vključuje naslednje podatke:

1. šifra zahtevka (`TransactionId`) – šifra transakcije kot jo določi odjemalec; podatek je obvezen in zagotavljanja revizijsko sled med Pladnjem in odjemalcem,
2. šifra/ime odjemalca (`AppId`) – šifra ali ime odjemalca; podatek je informativen,
3. čas zahtevka odjemalca (`SysTime`) – podatek je informativen za potrebe revizijske sledi.

Telo XML objekta (`B51BodyType`) vključuje naslednje podatke:

1. akcija – START (začetek zahtevka), STOP (konec zahtevka), CANCEL (preklic zahtevka),
2. seznam postopkov (`ProcedureId`) – šifra postopka/postopkov za katerega se izvaja pridobivanje podatkov; šifra postopka je namen zaradi katerega se izvaja poizvedovanje podatkov,
3. seznam emšo števil (`EmsList`) – emšo številke za katere se izvaja proces pridobivanja podatkov,
4. seznam davčnih števil (`DavStevList`) – davčne številke za katere se izvaja proces pridobivanja podatkov,
5. referenčni datum od (`RefDateFrom`), referenčni datum do (`RefDateTo`) – datuma za katero se izvaja poizvedovanje,

6. seznam param/value atributov (ParamValueList) – splošni parametri poizvedovanja (npr. MaticnaList – seznam matičnih števil, itd.

V nadaljevanju vidimo primer XML zahtevka za začetek izvajanja pridobivanja podatkov (<ns3:Action Id="START" Name="Začetek zahtevka"/>) za namen "ePostopek" in EMŠO 2309980504008:

```
<B51RequestType xmlns:ns2="http://nio.gov.si/names/pladenj/globaltypes"
xmlns:ns3="http://nio.gov.si/names/pladenj/b51">
  <ns3:Header>
    <ns3:ApplId Id="1" Name="eOdjemalec"/>
    <ns3:TransactionId Id="sifra_transakcije_03" Name="Številka vloge"/>
    <ns3:SysTime>2014-11-13T14:36:07.912Z</ns3:SysTime>
  </ns3:Header>
  <ns3:Body>
    <ns3:Action Id="START" Name="Začetek zahtevka"/>
    <ns3:ProcedureId Id="ePostopek"/>
    <ns3:Emsolist>
      <ns2:Emsos>2309980504008</ns2:Emsos>
    </ns3:Emsolist>
    <ns3:RefDateFrom>2014-11-13</ns3:RefDateFrom>
    <ns3:ParamValueList>
      <ns3:Param>DATUM_OD</ns3:Param>
      <ns3:Value>2014-01-01</ns3:Value>
    </ns3:ParamValueList>
  </ns3:Body>
</B51RequestType>
```

V sistemu Pladenj je postopek "ePostopek" administriran tako, da postopek vsebuje poizvedbe za en ali več virov podatkov. Poglavje 3.1.2 opisuje koncept postopka v sistemu Pladenj. Torej, odjemalec ne določi eksplicitno podatkovne vire, ampak postopek oz. namen za katerega Pladenj izvaja pridobivanje podatkov na osnovi določene zakonske podlage. Na ta način, odjemalec ne skrbi za seznam virov, ki jih je potrebno klicati, ker je ta seznam določen v namenu oz. postopku. Na primer, za potrebe odjemalca eSociala, Pladenj vzdržuje naslednje postopke oz. namene: otroški dodatek (OD), denarna pomoč (DP), državna štipendija (DS), znižano plačilo vrtca (VR), subvencija malice (MU), subvencija kosila (KU), subvencije prevozov (PD), subvencija plačila socialnovarstvenih storitev (SO), prispevek k plačilu družinskega pomočnika (DR), varstveni dodatek (VD), subvencija najemnine (NA), plačilo prispevka za obvezno zdravstveno zavarovanje (OZ), kritje do polne vrednosti zdravstvenih storitev (DZ), itd.

Sistem podpira tudi možnost vključevanja več postopkov v proces poizvedovanja (npr. <ns3:ProcedureId Id="OD"/><ns3:ProcedureId Id="DP"/>) in pri tem ugotovi unijo poizvedb za katere je potrebno izvesti pridobivanje podatkov. Na ta način ni potrebno večkrat klicati vir z istimi podatki.

Sistem podpira tudi atomarne postopke za katere se kliče samo en podatkovni vir (npr. <ns3:ProcedureId Id="eAJPES"/>) in je pridobivanje podatkov hitrejše, ker se za zahtevek ne sproži BPM proces, ampak se takoj izvede klic metode podatkovnega vira. Za atomarne postopke Pladenj ne zagotavlja funkcionalnost samo-ponovitve (t.i. auto-resume) tako, da je v primeru napake potrebno poklicati Pladenj še enkrat.

Metoda `handleRequest` izvaja naslednja splošna preverjanja vhodnih parametrov XML zahtevka:

1. "ERR_606_ODJEMALEC_NOT_FOUND_FOR_USER" - v Pladnju ne obstaja uporabnik za odjemalca v zahtevku.
2. "ERR_001_NO_CREDENTIALS" - odjemalec nima pravice za izvajanje.
3. "ERR_011_TRANSACTION_FINISHED" - številka zahtevka je že uporabljena in zaključena.

V primeru akcije START se preverja:

1. "ERR_014_MISSING_DATE_FROM" - obvezen referenčni datum.
2. "ERR_002_NO_PROCEDURE" - obvezna šifra postopka.
3. "ERR_013_INVALID_PROCEDURE_PARAMS" - v primeru kombiniranega postopka (več postopkov v zahtevku), atomarni in kompleksni postopki niso dovoljeni.
4. "ERR_005_WRONG_PROCEDURE" - napačen postopek v zahtevku.
5. "ERR_012_NO_REQUIRED_PARAMS" - zahtevek mora vključevati EMŠO številko, ali davčno številko.
6. "ERR_006_BAD_VAT_ID" - napačna davčna številka.
7. "ERR_007_BAD_EMSO" - napačna EMŠO številka.
8. "ERR_010_ILLEGAL_TRANS_POST" - preverjanje načina izvajanja (enkratno, večkratno).

V primeru akcija START in pravih vhodnih parametrov, Pladenj:

1. vrne odgovor odjemalcu `<xs:enumeration value="OK" />`
2. začne BPM proces pridobivanja podatkov od podatkovnih virov. Za atomarne postopke izvede samostojno poizvedbo pridobivanja podatkov od podatkovnega vira in vrne odgovor kot je predstavljene v poglavju 3.2 Spletna storitev B51CallackWs - obveščanje.

V primeru akcije STOP ali CANCEL in pravih vhodnih parametrov, Pladenj vrne odjemalcu odgovor `<xs:enumeration value="OK" />`, zaključi transakcijo, arhivira meta podatke in izbriše podatke. Pladenj preveri še:

1. "ERR_008_STOP_WITH_PROCEDURE" - zahtevek za STOP ne sme vključevati šifro postopka.
2. "ERR_009_CANCEL_WITH_PROCEDURE" - zahtevek za CANCEL ne sme vključevati šifro postopka.

V nadaljevanju so predstavljeni vsi možni statusi, ki jih Pladenj pošlje odjemalcu. Že prej omenjeni statusi se vrnejo ob klicu metode `handleRequest` spletne storitve B51Ws. Ostale statusse Pladenj pošlje odjemalcu ob klicu spletne storitve B51CallackWs.

```
<xs:simpleType name="StatusIdType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="OK" />
    <!-- 0xx,10x,11x general errors -->
    <xs:enumeration value="ERR_001_NO_CREDENTIALS" />
    <xs:enumeration value="ERR_002_NO_PROCEDURE" />
    <xs:enumeration value="ERR_003_NO_CODE_TYPE" />
    <xs:enumeration value="ERR_004_REQUEST_NOT_SENT" />
    <xs:enumeration value="ERR_005_WRONG_PROCEDURE" />
    <xs:enumeration value="ERR_006_BAD_VAT_ID" />
    <xs:enumeration value="ERR_007_BAD_EMSO" />
    <xs:enumeration value="ERR_008_STOP_WITH_PROCEDURE" />
    <xs:enumeration value="ERR_009_CANCEL_WITH_PROCEDURE" />
    <xs:enumeration value="ERR_010_ILLEGAL_TRANS_POST" />
    <xs:enumeration value="ERR_011_TRANSACTION_FINISHED" />
    <xs:enumeration value="ERR_012_NO_REQUIRED_PARAMS" />
    <xs:enumeration value="ERR_013_INVALID_PROCEDURE_PARAMS" />
```

```

<xs:enumeration value="ERR_101_RDBMS_INTERNAL" />
<xs:enumeration value="ERR_102_INTERNAL_ERROR" />
<xs:enumeration value="ERR_103_NO_DATASOURCE_RESULT" />
<xs:enumeration value="ERR_104_WRONG_PUBLIC_KEY" />
<xs:enumeration value="ERR_105_WRONG_TEMPLATE_DATA" />
<xs:enumeration value="ERR_106_WRONG_KEYSTORE" />
<xs:enumeration value="ERR_107_SOAP_CALL_EXCEPTION" />
<xs:enumeration value="ERR_108_REST_CALL_EXCEPTION" />
<xs:enumeration value="ERR_109_JMS_ERROR" />
<xs:enumeration value="ERR_110_NO_QUERY" />
<xs:enumeration value="ERR_111_JNDI_LOOKUP_ERROR"/>
<xs:enumeration value="ERR_120_BAD_DATA_IN_XML" />
<xs:enumeration value="ERR_121_BAD_DATA" />
<xs:enumeration value="ERR_122_JBPM_ERROR" />
<xs:enumeration value="ERR_123_NO_PROC_DEF"/>
<xs:enumeration value="ERR_124_JBPM_STOP_EXECUTION"/>
<xs:enumeration value="ERR_125_JBPM_FORK_ERROR"/>
<xs:enumeration value="ERR_126_JBPM_JOIN_ERROR"/>
<xs:enumeration value="ERR_201_JAXB_ERROR"/>
<xs:enumeration value="ERR_241_XPATH_FOUND_NO_RESULTS" />
<xs:enumeration value="ERR_400_DATA_NOT_FOUND" />
<xs:enumeration value="ERR_401_POSTOPEK_NOT_FOUND" />
<xs:enumeration value="ERR_402_TRANSACTION_NOT_FOUND" />
<xs:enumeration value="ERR_403_MONITOR_NOT_FOUND" />
<xs:enumeration value="ERR_404_POSTOPEKPOIZVEDBA_NOT_FOUND" />
<xs:enumeration value="ERR_405_OPERACIJA_POIZVEDBA_NOT_FOUND" />
<xs:enumeration value="ERR_406_DB_CONNECTIVITY"/>
<xs:enumeration value="ERR_501_INVALID_XML" />
<xs:enumeration value="ERR_502_CRYPT"/>
<xs:enumeration value="ERR_601_NO_SYSTEM_OPERATION_EXISTS" />
<xs:enumeration value="ERR_602_BAD_URL" />
<xs:enumeration value="ERR_603_POSTOPEK_HAS_NO_ODJEMALEC" />
<xs:enumeration value="ERR_604_FILE_NOT_FOUND"/>
<xs:enumeration value="ERR_605_UNKNOWN_GROOVY_TYPE"/>
<xs:enumeration value="ERR_606_ODJEMALEC_NOT_FOUND_FOR_USER"/>
<xs:enumeration value="ERR_014_MISSING_DATE_FROM"/>
<xs:enumeration value="INF_006_WAITING_RESPONSE" />
<xs:enumeration value="INF_404_WAITING_AUTOMATED_RESUME" />
<xs:enumeration value="WARN_000_GENERIC"/>
<xs:enumeration value="WARN_001_XML_INVALID"/>
</xs:restriction>
</xs:simpleType>

```

3.1.1.1 Sinhroni klic – metoda handleRequest3

Metoda `handleRequest3` omogoča sinhrono pridobivanje podatkov in zato podpira samo sinhron način komuniciranja pladnja z podatkovnimi viri. Metoda `handleRequest3` ima enake vhodne parametre kot metoda `handleRequest`, razlikuje se pa v odgovoru, kjer vrne enak odgovor kot če bi klicali metodo `getTransactionData`. V primeru kakršnekoli napake pri sinronem klicanju je poizvedovanje ustavljeno. Sinhroni klic vrne v primeru uspešnega klica direktno podatke(`getTransactionData`) ali napako.

Primer vhodnih parametrov za sinhron klic:

```
<B51RequestType xmlns:ns2="http://nio.gov.si/names/pladenj/globaltypes"
xmlns:ns3="http://nio.gov.si/names/pladenj/b51">
  <ns3:Header>
    <ns3:ApplId Id="1" Name="e0djemalec"/>
    <ns3:TransactionId Id="sifra_transakcije_03" Name="Številka vloge"/>
    <ns3:SysTime>2014-11-13T14:36:07.912Z</ns3:SysTime>
  </ns3:Header>
  <ns3:Body>
    <ns3:Action Id="START" Name="Začetek zahtevka"/>
    <ns3:ProcedureId Id="ePostopek"/>
    <ns3:EmsoList>
      <ns2:Emso>2309980504008</ns2:Emso>
    </ns3:EmsoList>
    <ns3:RefDateFrom>2014-11-13</ns3:RefDateFrom>
    <ns3:ParamValueList>
      <ns3:Param>DATUM_OD</ns3:Param>
      <ns3:Value>2014-01-01</ns3:Value>
    </ns3:ParamValueList>
  </ns3:Body>
</B51RequestType>
```

```
<B51ResponseType xmlns:ns2="http://nio.gov.si/names/pladenj/globaltypes"
xmlns:ns3="http://nio.gov.si/names/pladenj/b51">
  <ns3:Header>
    <ns3:AppId Id="1" Name="eOdjemalec"/>
    <ns3:TransactionId Id="sifra_transakcije_03"/>
    <ns3:SysTime>2014-11-13T16:41:30.211+01:00</ns3:SysTime>
  </ns3:Header>
  <ns3:Body>
    <ns3:Status Id="OK" Message="OK"/>
    <ns3:Action Id="START" Name="START"/>
    <ns3:ResponseList>
      <ns3:IdProcedure Id="ePostopek" Name="Pridobivanje podatkov o osebi po zakona xy"/>
      <ns3:DataSourceList>
        <ns3:Inst Id="INST1" Name="Institucija"/>
        <ns3:Source Id="1" Name="eVIR"/>
        <ns3:Operation Id="IO001" Name="Operacija za vir eVIR"/>
        <ns3:Query Id="1" Name="Poizvedba o osebi iz vira eVIR"/>
      </ns3:DataSourceList>
    </ns3:ResponseList>
  </ns3:Body>
</B51ResponseType>
```

Glede, da je šifra postopka najbolj pomemben parameter vhodnega XML zahtevka, je v tem poglavju natančneje opisan način kako sistem Pladenj vzdržuje postopke in kako kreira BPM procese za potrebe teh postopkov.

Pladenj omogoča gradnjo BPM procesov za potrebe postopkov in poizvedb v realnem času. Administracija postopkov določa pravila/navodila za dinamično orkestracijo procesov tako, da se v realnem času zgradi BPM proces za potrebe izvajanja zahtevka. Pri tej orkestraciji se upoštevajo vsa navodila iz administracije kot so poizvedbe in vrstni red izvajanja poizvedb, način izvajanja (paralelno, zaporedno), kombiniranje poizvedb iz različnih postopkov, shranjevanje pridobljenih podatkov na BPM kontekst in uporabljanje teh podatkov v naslednjih korakih BPM procesov, itd.

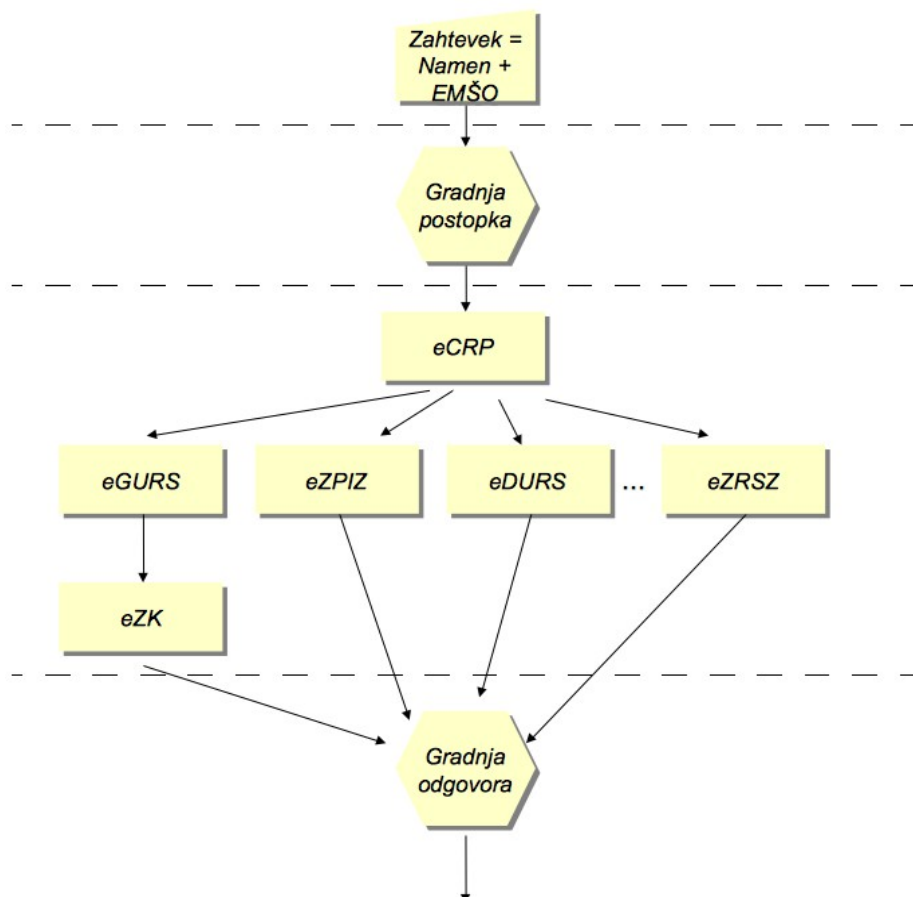
Na naslednji sliki vidimo primer procesa pridobivanja podatkov, ki vključuje vse faze izvajanja zahtevka.

V prvi fazi Pladenj preveri zahtevek in izlušči vse relevantne podatke.

V drugi fazi Pladenj zgradi BPM proces iz glavnega in množice mikro procesov, ki določajo obnašanje pridobivanja podatkov za določen podatkovni vir. Vsak pravokotnik predstavlja en mikro proces.

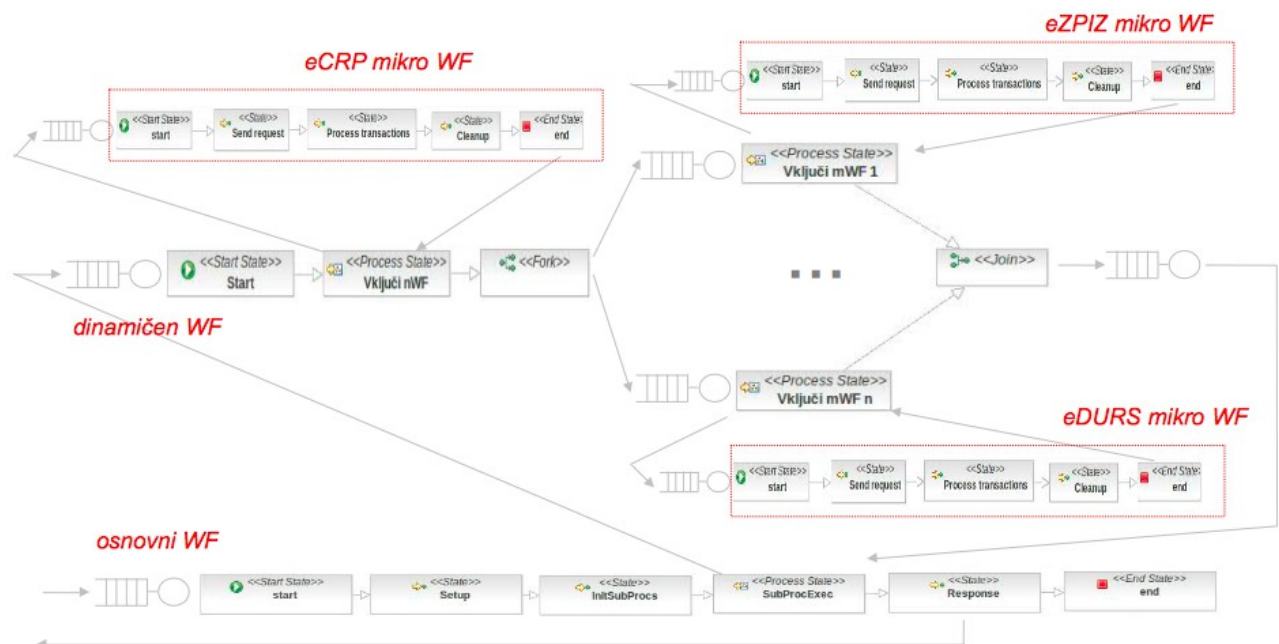
V tretji fazi procesa Pladenj začne orkestracijo dinamično zgrajenih mikro procesov. Na sliki vidimo, da proces v prvem koraku vključuje mikro proces za sistemsko poizvedbo iz vira CRP. Sistemska poizvedba se uporablja za pridobivanje prehodnih podatkov, kot so davčna, šifra občine, zakonci, itd. Sistemska poizvedba ne vrača odgovor odjemalcu in se ne shrani v transakcijsko zbirko. V drugem koraku faze, Pladenj paralelno izvaja mikro procese s poizvedbami za vsak posamezen podatkovni vir: eGURS, eZPIZ, eDURS, eZRSZ. Glede, da je vir eZK podatkovno odvisen od rezultatov odgovora vira eGURS, Pladenj ne izvede mikro proces za izvajanje poizvedbe za eZK dokler se ne konča izvajanje mikro procesa za eGURS.

V zadnji fazi se pridobljeni podatki združijo v XML in pošljejo odjemalcu.



Slika 5 - Faze izvajanja procesa

Kot je predstavljeno na sliki je vzporedno izvajanje poizvedb ena ključnih tehnoloških lastnosti Pladnja. Implementirana s pomočjo čakajočih vrst. Na ta način je možno izvajati veliko število paralelnih poizvedb (linearna rast sistema je omogočena z dodajanjem novih aplikacijskih strežnikov), kot je tudi možno omejevati število paralelnih poizvedb, da ne pride do prevelike obremenitve bodisi Pladnja ali podatkovnih virov. Zahtevek za spletno storitev za vsak podatkovni vir je možno poslati na ločeno čakajočo vrsto, kateri je možno določiti število paralelnih izvajanj in na ta način je možno uglaševati celotno delovanje sistema. Na naslednji sliki vidimo tehnološko implementacijo dinamične gradnje in orkestracije BPM procesov v sistemu Pladenj.



Slika 6 - Dinamična gradnja BPM procesov v realnem času

BPM proces na sliki predpostavlja, da je postopek sestavljen iz systemske eCRP poizvedbe (eCRP mikro WF) na prvem nivoju in več paralelnih poizvedb na drugem nivoju (eZPIZ mikro WF, itd.). Vidimo 3 vrste procesov:

1. Osnovni WF – statični BPM proces, ki ima vlogo priprave okolja za dinamično združevanje procesov. Ključna funkcionalnost osnovnega procesa je izvajanje v vozlišču „SubProcExec“, ko se v realnem času v tem vozlišču vgrajuje en ali več dinamičnih procesov. Druga pomembna funkcionalnost osnovnega procesa je v vozlišču „Response“, ko se zgradi XML odgovor in pošlje na spletni servis odjemalca.
2. Mikro WF – množica pred definiranih statičnih mikro procesov za vsak podatkovni vir posebej. V najslabšem primeru je potrebno implementirati toliko mikro procesov kot je podatkovnih virov. V večini primerih podatkovni viri izvajajo isto funkcionalnost (klic spletne storitve, obdelava odgovora, shranjevanje odgovora) tako, da je zelo različnih vrst mikro procesov malo. Funkcionalnost mikro procesa vključuje klic spletne storitve, obdelava odgovora (XML validacija, XPATH izločanje potrebnih podatkov, preverjanje podatkov za ponovno klicanje), kriptiranje in shranjevanje XML odgovora v bazo.
3. Dinamičen WF - vloga dinamičnega procesa je da združi mikro procese. Funkcionalnost dinamičnega procesa vključuje 3 osnovna vozlišča, ki se lahko večkrat ponavljajo. Vozlišče „Vključ mWF“ omogoča, da se v tej točki dinamično doda mikroWF. Vozlišče „Fork“ omogoča, da se v tej točki začne paralelizacija več mikro WF-jev. Vozlišče „Join“ omogoča združevanje paralelnih mikro WF-jev in čakanje na zadnji mikroWF v izvajanju. Dinamičen WF se zgradi na osnovi informacij iz podatkovne baze (kateri poizvedbe so v postopku, način izvajanja, vrstni red)

Kot je razidno na sliki je pred vsakim tipom procesa (osnovniWF, mikroWF, dinamičenWF) določena čakajoča vrsta, ki določa število paralelnih niti izvajanja. Na ta način je možno prilagajati delovanje sistema. Npr., če je v vrsti za osnovniWF določeno 5 niti je s tem omejeno 5 sočasnih zahtevkov. V primeru, da je več zahtevkov, kot je niti izvajanja, zahtevki čakajo v vrsti. Če je za mikroWF določeno 100 niti izvajanja, je v sistemu omogočeno 100 sočasnih poizvedovanj na podatkovne vire. V primeru, da je več poizvedb, te poizvedbe čakajo v vrsti. Čakajoče vrste zagotavljajo optimalno prepustnost sistema in jih je možno administrirati preko grafičnega vmesnika.

Na naslednji sliki vidimo grafični vmesnik, ki prikazuje postopke in pripadajoče poizvedbe. Na primer, postopek eMRVL je atomaren postopek in vključuje samo eno poizvedbo. Postopek OD je kompleksen in vključuje veliko število poizvedb.

Administracija → Postopki							
ID	ŠIFRA	NAZIV	POIZVEDBA	ODJEMALEC	NAČIN IZVAJANJA	VKLUČI	
1	eCRP	Atomaren postopek za CRP	1 - Pridobivanje podatkov iz zbirke CRP za EMŠO	1 - eSociala	Izvajanje določeno na odjemalcu	☐	☐
2	eGSP	Atomaren postopek za GSP	1 - Pridobivanje podatkov iz zbirke GSP za EMŠO	1 - eSociala	Izvajanje določeno na odjemalcu	☐	☐
3	eMRVL	Atomaren postopek za MRVL	1 - Pridobivanje podatkov iz zbirke MRVL za EMŠO	1 - eSociala	Izvajanje določeno na odjemalcu	☐	☐
4	eURSP	Atomaren postopek za URSP	1 - Pridobivanje podatkov iz zbirke URSP za EMŠO	1 - eSociala	Izvajanje določeno na odjemalcu	☐	☐
5	eDURS	Atomaren postopek za DURS	1 - Pridobivanje podatkov iz zbirke DURS za EMŠO	1 - eSociala	Izvajanje določeno na odjemalcu	☐	☐
6	eAJPES	Atomaren postopek za AJPES	1 - Pridobivanje podatkov iz zbirke CRP za EMŠO 1 - RTR pridobivanje podatkov iz zbirke AJPES	1 - eSociala	Izvajanje določeno na odjemalcu	☐	☐
7	eKDD	Atomaren postopek za KDD	1 - Pridobivanje podatkov iz zbirke KDD za EMŠO	1 - eSociala	Izvajanje določeno na odjemalcu	☐	☐
8	eJPRS	Atomaren postopek za JPRS	1 - Pridobivanje podatkov iz zbirke JPRS za EMŠO	1 - eSociala	Izvajanje določeno na odjemalcu	☐	☐
9	OD	Otroški dodatek	1 - Sistemsko pridobivanje podatkov iz zbirke CRP za EMŠO 1 - Pridobivanje podatkov o delih stavb iz zbirke GURS za EMŠO 1 - Pridobivanje podatkov iz zbirke AM banke 1 - Pridobivanje podatkov iz zbirke MRVL za EMŠO 1 - Pridobivanje podatkov iz zbirke ZRSZ za EMŠO 1 - Pridobivanje podatkov iz zbirke JPRS za EMŠO 1 - Pridobivanje podatkov iz zbirke ZZS za EMŠO 1 - Pridobivanje podatkov iz zbirke MSSSTAT 1 - Pridobivanje podatkov iz zbirke KDD za EMŠO 1 - Pridobivanje podatkov iz zbirke GSP za EMŠO 1 - Pridobivanje podatkov iz zbirke ZPIZ za EMŠO 1 - RTR pridobivanje podatkov iz zbirke AJPES 1 - RTR pridobivanje podatkov iz zbirke AJPES	1 - eSociala	Izvajanje določeno na odjemalcu	☒	☐

Slika 7 - Primeri postopkov

Vsak postopek ima vrstni red izvajanja poizvedb. Na naslednji maski vidimo postopek, ki vključuje več poizvedb. Poizvedba eCRP se izvaja prva (vrstni red = 1) in, ko je pridobitev podatkov za to poizvedbo končan se začne paralelno izvajanje pridobivanja podatkov preostalih poizvedb (vrstni red = 2).

Postopki: Določitev poizvedb				
ŠIFRA	NAZIV	PODRBNOСТИ	VRSTNI	
1	Sistemsko pridobivanje podatkov iz zbirke CRP za EMŠO 10000 - Operacija za login ↳ INSTITUCIJA: 6 - Ministrstvo za notranje zadeve, VIR: 1 - eCRP 10001 - Sistemsko pridobivanje podatkov CRP ↳ INSTITUCIJA: 6 - Ministrstvo za notranje zadeve, VIR: 1 - eCRP		1	
1	Pridobivanje podatkov o delih stavb iz zbirke GURS za EMŠO GURS01 - WfsGetFeatureVmiDSt ↳ INSTITUCIJA: 9 - Geodetska uprava Republike Slovenije, VIR: 2 - eGURS_		2	
1	Pridobivanje podatkov iz zbirke AM banke BANK_01 - Spletni servis AM banke ↳ INSTITUCIJA: 8 - Asinhroni modul, VIR: 3 - eBANKE		2	
1	Pridobivanje podatkov iz zbirke MRVL za EMŠO 2 - Operacija za pridobitev podatkov MRVL ↳ INSTITUCIJA: 16 - Ministrstvo za promet, VIR: 1 - eMRVL		2	
1	Pridobivanje podatkov iz zbirke ZRSZ za EMŠO ZRSZ_01 - Prijava v ZRSZ ↳ INSTITUCIJA: 10 - Zavod Republike Slovenije za zaposlovanje, VIR: 1 - e- ZRSZ_02 - Pridobivanje podatkov iz ZRSZ ↳ INSTITUCIJA: 10 - Zavod Republike Slovenije za zaposlovanje, VIR: 1 - e-		2	
1	Pridobivanje podatkov iz zbirke JPRS za EMŠO 2 - Operacija za pridobitev podatkov JPRS ↳ INSTITUCIJA: 2 - Javni jamstveni in preživninski sklad RS, VIR: 1 - eJPRS		2	
1	Pridobivanje podatkov iz zbirke ZZS za EMŠO 2 - Podatkovna operacija za vir ZZS ↳ INSTITUCIJA: 11 - Zavod za zdravstveno zavarovanje Slovenije, VIR: 1 -		2	
1	Pridobivanje podatkov iz zbirke MSSSTAT 2 - Operacija za pridobivanje podatkov MSSSTAT ↳ INSTITUCIJA: 15 - Ministrstvo za šolstvo in šport, VIR: 1 - eMSSSTAT		2	

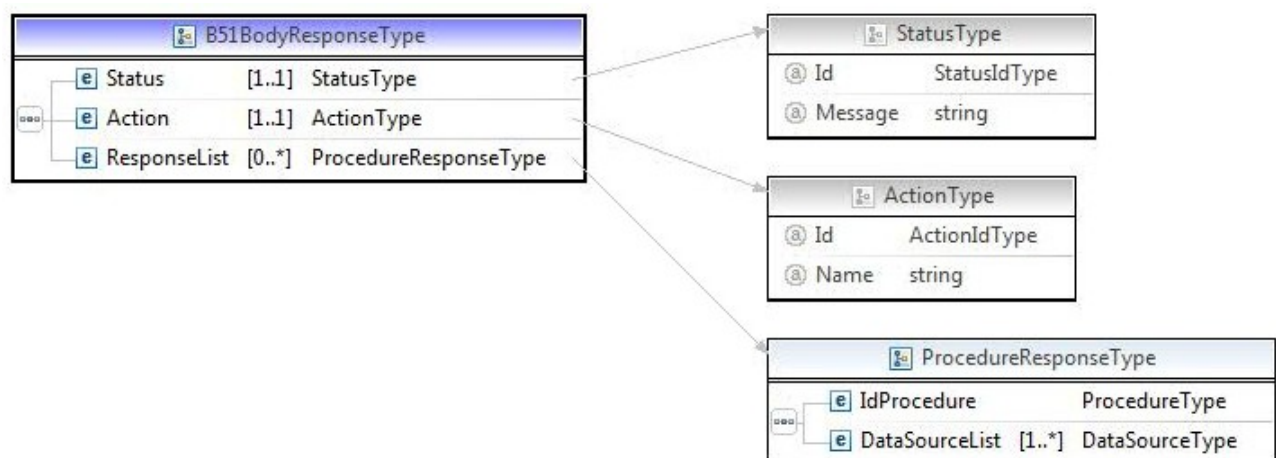
Slika 8 - Poizvedbe na postopku

3.2 Pridobivanje podatkov – spletna storitev B51Ws

Metoda `getTransactionData` vrača podatke pridobljene od podatkovnih virov v obliki XML ovojnice. Vhodni parameter metode je podatek tipa `TransactionType`. Podatek predstavlja šifro transakcije s katero se je sprožilo pridobivanje podatkov v metodi `handleRequest` spletne storitve B51Ws:

```
<TransactionType Id="sifra_transakcije_03" Name="Številka vloge"
xmlns:ns2="http://nio.gov.si/names/pladenj/globaltypes"/>.
```

Metoda je sinhrona in vrne odgovor tipa `B51ResponseType`. Spletna storitev preveri, če odjemalec kliče metodo s pravilnim digitalnim potrdilom in pravilno vlogo. Glava odgovora vključuje šifro zahtevka (`TransactionId`), šifro odjemalca (`ApplId`) in čas zahtevka odjemalca (`SysTime`), podobno kot za `B51BodyRequestType`. Telo odgovora (`B51BodyResponseType`) vsebuje globalni status (`StatusType`) in odgovor za vsak podatkovni vir (`ProcedureResponseType`).



Slika 9 - XSD struktura B51BodyResponseType

Status `<xs:enumeration value="OK" />` pomeni, da so vsi podatki uspešno pridobljeni od virov.

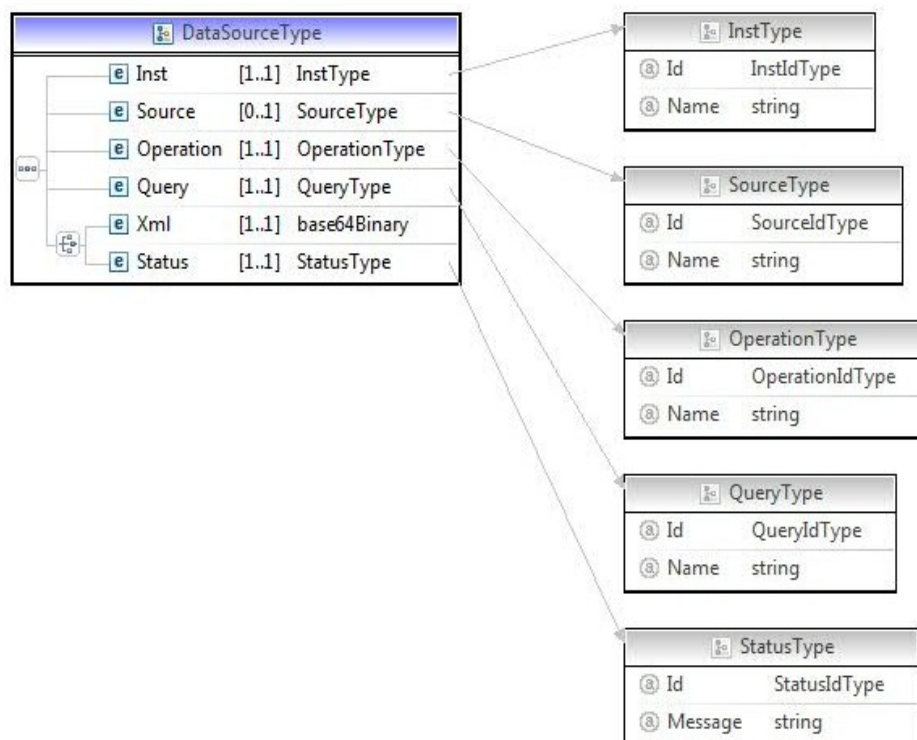
Status `<xs:enumeration value="ERR_102_INTERNAL_ERROR" />` pomeni, da je pridobivanje podatkov neuspešno za eno ali več poizvedb v postopku. V tem primeru, Pladenj je končal s poskusi samo-ponovitve (auto-resume) in je ustavil proces pridobivanja podatkov za neuspešno poizvedbo. Istočasno Pladenj obvesti administratorja o težavah pri izvajanju poizvedbe.

Status `<xs:enumeration value="INF_006_WAITING_RESPONSE"/>` pomeni, da Pladenj še čaka na odgovore določenih virov (npr. viri tipa asinhroni, paketni, pooling).

Status `<xs:enumeration value="INF_404_WAITING_AUTOMATED_RESUME" />` pomeni, da je pridobivanje podatkov neuspešno za eno ali več od poizvedb v postopku. Pladenj izvaja samo-ponovitev (auto-resume). Status se zgodi v primeru začasno nedosegljivega vira, napake v protokolu komunikacije z viri, napake na viru, sistemske napake na omrežju ali strojni opremi, napake v delovanju sistema Pladenj, itd. Ob tem statusu je velika verjetnost, da bo v določen času Pladenj pridobil podatke in vrnil status `<xs:enumeration value="OK" />`.

Status `<xs:enumeration value="WARN_001_XML_INVALID"/>` pomeni, da je en od virov vrnil odgovor v obliki, ki ne ustreza pričakovani XSD shemi odgovora.

Atribut `ProcedureResponseType` je ovojnica odgovorov od podatkovnih virov (`DataSourceType`) za določen postopek (`ProcedureType`). Vsebuje en atribut tipa `<ns3:IdProcedure/>` in en ali več atributov tipa `<ns3:DataSourceList>`, ki vsebuje meta podatke o viru ter XML odgovor od vira.



Slika 10 - XSD struktura DataSourceType

Atributi InstType, SourceType, OperationType in QueryType so meta podatki, ki opisujejo podatkovni vir, atribut Xml je odgovor podatkovnega vira, kodiran z javnim ključem odjemalca:

1. Institucija (`<ns3:Inst Id="INST1" Name="Institucija"/>`) - šifra in naziv institucije pri kateri je postavljen podatkovni vir.
2. Vir (`<ns3:Source Id="1" Name="eVIR"/>`) - šifra in naziv vira spletne storitve za pridobivanje podatkov; ena institucija ima več različnih virov za pridobivanje podatkov.
3. Operacija (`<ns3:Operation Id="I0001" Name="Operacija za vir eVIR"/>`) - šifra in naziv operacije s katero se kliče metoda spletne storitve.
4. Poizvedba (`<ns3:Query Id="1" Name="Poizvedba o osebi iz vira eVIR"/>`) - enota pridobivanja podatkov, npr. ena poizvedba vključuje več operacij (login, getData, logout).
5. Odgovor (`<ns3:Xml>`) - kodiran odgovor.
6. Status (`<ns3:Status Id="OK" Message="OK"/>`) - status uspešnosti pridobivanja podatka za vir.

V nadaljevanju vidimo primer XML ovojnice odgovora, ki jo vrne metoda `getTransactionData`:

```
<B51ResponseType xmlns:ns2="http://nio.gov.si/names/pladenj/globaltypes"
xmlns:ns3="http://nio.gov.si/names/pladenj/b51">
  <ns3:Header>
    <ns3:ApplId Id="1" Name="eOdjemalec"/>
    <ns3:TransactionId Id="sifra_transakcije_03"/>
    <ns3:SysTime>2014-11-13T16:41:30.211+01:00</ns3:SysTime>
  </ns3:Header>
  <ns3:Body>
```

```

<ns3:Status Id="OK" Message="OK"/>
<ns3:Action Id="START" Name="START"/>
<ns3:ResponseList>
  <ns3:IdProcedure Id="ePostopek" Name="Pridobivanje podatkov o osebi po zakona xy"/>
  <ns3:DataSourceList>
    <ns3:Inst Id="INST1" Name="Institucija"/>
    <ns3:Source Id="1" Name="eVIR"/>
    <ns3:Operation Id="IO001" Name="Operacija za vir eVIR"/>
    <ns3:Query Id="1" Name="Poizvedba o osebi iz vira eVIR"/>
  </ns3:DataSourceList>
</ns3:ResponseList>
</ns3:Body>
</B51ResponseType>

```

Odjemalec metodo `getTransactionData` lahko kliče neomejeno krat. Tudi če ni dobil povratno informacijo od Pladnja o uspešnem koncu pridobivanja podatkov od virov. V tem primeru bo v XML odgovoru verjetno več virov v statusu `<xs:enumeration value="ERR_102_INTERNAL_ERROR" />`, ker Pladenj še ni končal izvajanje procesa in so poizvedbe v statusu napak. Pravilen protokol komunikacije med odjemalcem in Pladnjem je tak, da odjemalec pokliče metodo `getTransactionData` takrat, ko dobi obvestilo od Pladnja o koncu pridobivanja podatkov. To obvestilo se zgodi, ko Pladenj pokliče metodo `handleResponse` spletne storitve `B51CallbackWs` odjemalca.

3.2.1 Dekriptiranje pridobljenih podatkov

Vsak posamezen odgovor, ki ga pladenj vrne, je kriptiran in Base64 kodiran. Pladenj zakriptira xml odgovor vira z simetričnim kriptirnim algoritmom AES. Za ta algoritem najprej zgenerira simetričen ključ dolžine 128 bito, kriptiranemu dokumentu pa nato doda informacijo o simetričnem ključu, ke je še dodanto kriptiran z asimetričnim javnim ključem RSA. To zagotavlja da lahko podatke odkriptira samo uporabnik ki poseduje ustrezen privatni ključ.

Javanski primer kode za enkripcijo:

```
public static String encrypt(String xml, Key keyEncryptKey) throws Exception
{
    // Pretvorba vhodnega xml odpora v string obliki v DOM objekt
    org.w3c.dom.Document document = parseXmlToDocument(xml);

    // generiranje 128 bitnega AES simetričnega ključa
    java.security.Key symmetricKey = XmlKeyTool.GenerateSymmetricKey();

    // kriptiranje simetričnega ključa z javnim RSA ključem
    org.apache.xml.security.encryption.XMLCipher keyCipher =
    XMLCipher.getInstance(XMLCipher.RSA_OAEP);
    keyCipher.init(XMLCipher.WRAP_MODE, keyEncryptKey);
    org.apache.xml.security.encryption.EncryptedKey encryptedKey =
    keyCipher.encryptKey(document, symmetricKey);

    // definiramo element v XML-ju, ki ga želimo kriptirati (kriptiramo cel dokument, zato
    // izberemo root element)
    org.w3c.dom.Element rootElement = document.getDocumentElement();
    org.w3c.dom.Element elementToEncrypt = rootElement;
    boolean encryptContentsOnly = false;

    // inicializacija XMLChiper-ja s simetričnim ključem
    org.apache.xml.security.encryption.XMLCipher xmlCipher =
    XMLCipher.getInstance(XMLCipher.AES_128);
    xmlCipher.init(XMLCipher.ENCRYPT_MODE, symmetricKey);

    // dodajanje informacije o kriptiranem ključu k xml dokumentu
    org.apache.xml.security.encryption.EncryptedData encryptedDataElement =
    xmlCipher.getEncryptedData();
    KeyInfo keyInfo = new KeyInfo(document);
    keyInfo.add(encryptedKey);
    encryptedDataElement.setKeyInfo(keyInfo);

    // dejansko kriptiranje
    xmlCipher.doFinal(document, elementToEncrypt, encryptContentsOnly);

    // transformacija XML Document-a nazaj v String
    javax.xml.transform.TransformerFactory transformerFactory =
    TransformerFactory.newInstance();
    javax.xml.transform.Transformer transformer = transformerFactory.newTransformer();
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");

    javax.xml.transform.stream.StreamResult result = new StreamResult(new StringWriter());
    javax.xml.transform.dom.DOMSource source = new DOMSource(document);
    transformer.transform(source, result);

    String xmlString = result.getWriter().toString();

    return xmlString;
}
```

Javanski primer kode za dekapicijo:

```
public static String decrypt(String encryptedXml) throws Exception
{
    org.apache.xml.security.Init.init();
    //Pretvorba xmlodgovora v string obliki v DOM objekt
    org.w3c.dom.Document document = parseXmlToDocument(encryptedXml);

    String namespaceURI = "http://www.w3.org/2001/04/xmlenc#";
    String localName = "EncryptedData";

    //nalaganje privatnega ključa
    java.security.PrivateKey pk = loadPrivateKey();
    byte[] privateKeyRSA = pk.getEncoded();

    java.security.spec.PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(privateKeyRSA);
    java.security.KeyFactory kf = KeyFactory.getInstance("RSA");
    java.security.Key keyEncryptKey = kf.generatePrivate(spec);

    org.apache.xml.security.encryption.XMLCipher xmlCipher = XMLCipher.getInstance();
    xmlCipher.init(XMLCipher.DECRYPT_MODE, null);
    xmlCipher.setKEK(keyEncryptKey);

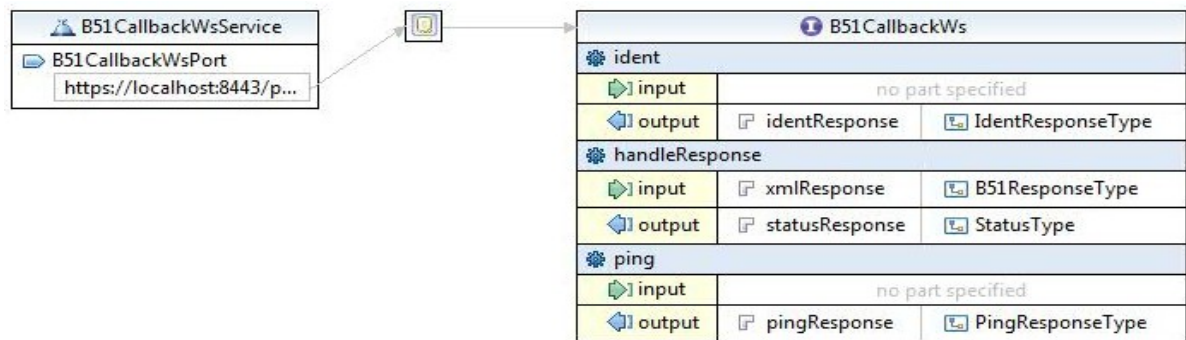
    org.w3c.dom.Element encryptedDataElement = (Element)
    document.getElementsByTagNameNS(namespaceURI, localName).item(0);
    //Dekriptiranje
    xmlCipher.doFinal(document, encryptedDataElement);

    //Pretvorba iz DOM objekta v
    String receivedXML = parseXmlDocumentToString(document);
    return receivedXML;
}
```

3.3 Obveščanje – spletna storitev B51CallbackWs

Na koncu BPM procesa pridobivanja podatkov od virov, Pladenj pokliče metodo spletne storitve odjemalca in posreduje XML odgovor o statusu pridobljenih podatkov iz podatkovnih virov. Odgovor ne vsebuje XML odgovore od podatkovnih virov, ampak samo meta podatke, informacijo o uspešnosti oz. neuspešnosti pridobivanja podatkov, ter razlog za morebitno napako.

Odjemalec implementira spletno storitev na osnovi B51CallbackWs.wsdl. Klic spletne storitev je omogočen aplikacijam s kvalificiranim digitalnim potrdilom. Torej, za klic metod spletnega servisa B51CallbackWs je potrebno pridobiti certifikat od sistema Pladenj in ga administrirati v varnostno shemo sistema odjemalca.



Slika 11 - WSDL za spletno storitev B51CallbackWs

Metoda `handleResponse` je sinhrona in vrne status o uspešnem prejemu klica, npr:

```
ns1:handleResponseResponse xmlns:ns1="http://nio.gov.si/names/pladenj/entrypoint">
  <statusResponse Id="OK" xmlns:ns2="http://nio.gov.si/names/pladenj/globaltypes"
  xmlns:ns3="http://nio.gov.si/names/pladenj/b51"/>
</ns1:handleResponseResponse>
```

V nadaljevanju vidimo primer XML-ja, ki ga Pladenj pošlje kot parameter v metodo `handleResponse` spletne storitve `B51CallbackWs`:

```
<B51ResponseType xmlns:ns2="http://nio.gov.si/names/pladenj/globaltypes"
xmlns:ns3="http://nio.gov.si/names/pladenj/b51">
  <ns3:Header>
    <ns3:ApplId Id="1" Name="eOdjemalec"/>
    <ns3:TransactionId Id="sifra_transakcije_03"/>
    <ns3:SysTime>2014-11-13T16:41:30.211+01:00</ns3:SysTime>
  </ns3:Header>
  <ns3:Body>
    <ns3:Status Id="OK" Message="OK"/>
    <ns3:Action Id="START" Name="START"/>
    <ns3:ResponseList>
      <ns3:IdProcedure Id="ePostopek" Name="Pridobivanje podatkov o osebi po zakona xy"/>
      <ns3:DataSourceList>
        <ns3:Inst Id="INST1" Name="Institucija"/>
        <ns3:Source Id="1" Name="eVIR"/>
        <ns3:Operation Id="IO001" Name="Operacija za vir eVIR"/>
        <ns3:Query Id="1" Name="Poizvedba o osebi iz vira eVIR"/>
        <ns3>Status Id="OK" Message="OK"/>
      </ns3:DataSourceList>
    </ns3:ResponseList>
  </ns3:Body>
</B51ResponseType>
```

Vidimo, da je vsebina XML-ja identična kot v primeru klica metode `getTransacitonData` s to razliko, da v ovojnici XML-ja ni atributa `<ns3:Xml>`.

Pladenj lahko pošlje odjemalcu delne odgovore. Na primer, če je določen vir nedosegljiv bo Pladenj poskušal izvajati funkcijo samo-ponovitve (auto-resume) in v tem primeru bo vrnil naslednji XML:

```
<B51ResponseType xmlns:ns2="http://nio.gov.si/names/pladenj/globaltypes"
xmlns:ns3="http://nio.gov.si/names/pladenj/b51">
  <ns3:Header>
    <ns3:ApplId Id="1" Name="eOdjemalec"/>
```

```

    <ns3:TransactionId Id="sifra_transakcije_03"/>
    <ns3:SysTime>2014-11-13T16:41:30.211+01:00</ns3:SysTime>
  </ns3:Header>
  <ns3:Body>
    <ns2:Status Id="INF_404_WAITING_AUTOMATED_RESUME" Message="Auto-resume poizvedba"/>
    <ns3:Action Id="START" Name="START"/>
    <ns3:ResponseList>
      <ns3:IdProcedure Id="ePostopek" Name="Pridobivanje podatkov o osebi po zakona xy"/>
      <ns3:DataSourceList>
        <ns3:Inst Id="INST1" Name="Institucija"/>
        <ns3:Source Id="1" Name="eVIR"/>
        <ns3:Operation Id="IO001" Name="Operacija za vir eVIR"/>
        <ns3:Query Id="1" Name="Poizvedba o osebi iz vira eVIR"/>
        <ns2:Status Id="INF_404_WAITING_AUTOMATED_RESUME" Message="Auto-resume poizvedba"/>
      </ns3:DataSourceList>
    </ns3:ResponseList>
  </ns3:Body>
</B51ResponseType>

```

V primeru asinhronih virov, ko Pladenj v status zapiše:

```

<ns2:Status Id="INF_006_WAITING_RESPONSE" Message="INF_006_WAITING_RESPONSE"/>.

```

V primeru, da odjemalec ne vrne status OK (<statusResponse Id="OK">), Pladenj sproži funkcionalnost samoponovitve z intervali, kot je določeno z administrativnimi parametri.

3.4 Zaključek postopka – spletna storitev B51Ws

Metoda `handleRequest` spletne storitve B51Ws se uporablja tudi za zaključek postopka izvajanja pridobivanja podatkov od virov. Odjemalec zgradi vhodni XML tipa `B51RequestType`, ki vsebuje akcijo STOP in šifro transakcije za katero se izvede zaključevanje procesa:

```

<B51RequestType xmlns:ns2="http://nio.gov.si/names/pladenj/globaltypes"
xmlns:ns3="http://nio.gov.si/names/pladenj/b51">
  <ns3:Header>
    <ns3:ApplId Id="eOdjemalec"/>
    <ns3:TransactionId Id="sifra_transakcije_03" Name="Številka vloge"/>
    <ns3:SysTime>2014-11-13T15:39:17.712Z</ns3:SysTime>
  </ns3:Header>
  <ns3:Body>
    <ns3:Action Id="STOP" Name="Začetek zahtevka"/>
  </ns3:Body>
</B51RequestType>

```

Ob zaključevanju procesa se vsi meta podatki postopka shranijo v arhivsko bazo. Iz transakcijske baze Pladenj se zbršejo vse informacije za zaključen postopek. Pridobivanje podatkov za to šifro transakcije ni več možno.

4 VKLJUČEVANJE NOVEGA ODJEMALCA NA PLADENJ

Zahteva za odjemalce:

Zahteve	Da/Ne
Implementacija B51CallbackWs spletne storitvenega	
Naslov B51CallbackWs	
Javni ključ odjemalske aplikacije	
Javni ključ pladnja do odjemalca	
Naslov B51Ws	
Opis vhodnih parametrov	

Zahteve za MJU:

[illegible]