INFOSYS - UŽICE

ИНФО АПИ СЕРВЕР В1

ПРЕГЛЕД РАЗЛИКА ИЗУМЕЂУ ПРОТОКОЛА ВЕРЗИЈЕ 0 И ВЕРЗИЈЕ 1

верзија 1.00

Пеђа Супуровић
Ужице, септембар 2020.

УВОД

Инфо АПИ сервер је универзалан модул који омогућава приступ Инфо програму и бази података удаљено, преко веб сервиса за све врсте клијената.

Од новембра 2017. у употербу је уведен ИнфоАПИ протокол верзије 1. он је увео углавном знатно већу контролу исправности рада клијентских аплиакција, исправности података и повећану безбедност података.

Уколико сте направили или правите АПИ клијент по протоколу верзије 1. ово објашњење се на вас не односи. Намењено је програмерима који су раније направили АПИ клијенте по верзији 0 ради евентуалног усаглашавања са верзијом 1.

Разлике између протокола нису велике и требало би да буде релативно једноставно унапредити клијенте који су нарпављени за ИнфоАПИ верзије 0 да раде са верзијом 1.

У овом тексту ћемо објаснити најважније разлике.

У тексту се користе цитати из Упутства за употребу АПИ функција које је посебан документ са детаљним објашњењиам протокола верзије 1. С временом, је могуће да ће Упутство бити допуњавано. Уколико постоји разлика у тексту у овом документу и у Упутству, примењује се онако како стоји у Упутству. Овај текст вас само упућује на битне разлике између верзије 0 и верзије 1 на које треба обратити пажњу. Примена протокола се врши према Упутству.

Садржај упита

ИнфоАПИ комуникација је знатно поједностављена верзија *RESTful* протокола. Разлика је у томе што се не употребљавају идентификатори ресурса, и употребљава се само ХТТП метод. Сви потребни подаци се налазе у *JSON* поруци.

HTTP упитом се подаци прослеђују *POST* методом, који садржи следеће променљиве:

json (char)	*С*адржи АПИ упит у *JSON* формату према спецификацији ИнфоАПИ сервера.
debuglevel (int)	Ниво дибаг логовања. Ако дибаг лог није потребан, треба да буде нула (или празно)
debugsection (char)	Листа дибаг секција које треба укључити у лог. Ако дибаг није потребан, треба да буде празно.
apiversion (char)	Ознака верзије АПИ-ја коју треба покренути за дати упит. Клијенти који раде по протоколу верзије 1 или новије треба да пошаљу верзију протокола или, боље, верзију АПИ-ја на којој је клијент развијан (на пример 1.18).

Овде је новина параметар apiversion. У верзији 0 он није постојао. У верзији 1 је он обавезан.

TAPIQUERYENVELOPE

Оквир (*TAPIQueryEnvelope*) је универзалан део JSON упита који је непроменљив без обзира на врсту упита. Његова улога је да обезбеди комуникацију између клијента и АПИ сервера. Он у себи садржи угњежден слој који садржи параметре позваној АПИ функцији.

НАвешћемо битне промене у структури:

req_id (char)	Идентификатор упита. Овај податак генерише клијент а сервер га враћа у одговору. Служи да клијент може да по потреби повеже одговоре са упитима и за рачунање контролног кода за проверу целевитости послатог упита. Клијент треба да употребљава различиту вредност за сваки упит који шаље. Ово је обавезан податак.

Овај податак постоји и у верзији нула али није постојао јасан захтев да идентификатор упита мора да буде различит за сваки упит.

api_version (char)	Ознака верзије АПИ-ја који клијент разуме. Навести конкретну верзију на којој је клијент развијан, на пример 1.18)

Овај податак је уведен у верзији 1. У њему треба навести захтевану верзију ИнфоАПИ-ја, исто као у параметрима ХТТП упита.

checksum (char)	Контролни код поруке. Намена му је да се преко њега може проверити интегритет послатог упита. Ово је обавезан податак.

У верзији 1 је уведен контролни код поруке као обавезан податак.

TAPIPARAMS

database_format (char)	Назнака формата у коме се налазе табеле у пољу *databaseattachment*. За сада су имплементирана три формата: *PLAIN_OBJ* – табеле су објекти који се преносе као *JSON* стринг *PLAIN_OBJ_BASE64* – табеле су објекти, а шаљу се кодирани (последица је крађи JSON) NATIVE_*OBJ* – табеле су објекти *SQLITE* – табеле су смештене у *SQLite* базу ZIPSQLITE – табеле су смештене у *SQLite* базу а onda kompresovane u ZIP arhivu ZIPDBF – табеле су смештене у *ZIP* архиву као фокспро табеле Ако није наведен формат (поље је празно) подразумева се *PLAIN_OBJ*. Клијентске апликације су обавезне да подрже формате *PLAIN_OBJ* и *SQLITE*.
result_database_format (char)	Опис формата који треба применити за враћање табеларних података као резултата извршења функције. Погледати опис поља *database_format*.

У верзији 1 су додати неки нови формати који су на располагању. Могу се употребљавати по потреби.

TAPIRESULT

exec_date_time (datetime)	Датум и време када је АПИ функција извршена на серверу.

Овај податак је додат. Клијент може да га употеби ако му је потребно да се веже за време извршења неке радње на серверу.

ОБРАДА ГРЕШАКА

У верзији 1 је уведен је изричит протокол за обраду грешака како је овде објашњено. Он се не разликује много у односу на ранији и компатибилан је, само су сада ствари боље утврђене и објашњене.

Након извршења, свака АПИ функција враћа статус о грешци. Подразумевани статус је да је функција извршена без грешака и да је вратила очекивани резултат.

Уколико дође до било које грешке АПИ функција ће вратити статус грешке чиме ће назначити да је дошло до грешке приликом извршења. Ако се ради о опису грешке коју враћа сама АПИ функција, те грешке ће бити документоване у опису функције.

Клијентска апликација треба да реагује на ове статусе. Подразумевани принцип је, да ако је дошло до неке грешке, клијентска апликација треба да сматра да позвана АПИ функција није урадила обраду која је очекивана нити да је вратила резултате који су очекивани и не сме да настави рад као да грешке није било. Она треба да обавести корисника о насталој грешци (или да је забележи у лог) и не треба да понавља исти позив функцији.

Уколико клијентска апликација познаје статусе грешака, она може на њих да реагује и другачије, односно на начин који следи из статуса грешке.

ТИПОВИ СТАТУСА ГРЕШАКА

Статус типа ОБАВЕШТЕЊЕ

Овај статус клијенту назначава да је настао неки проблем приликом извршења фунције али да он није изазвао прекид рада функције. Обично се тако може пријавити да је неки улазни податак недостајао или је био нетачан али да је АПИ функција сама то на неки начин регулисала, тако да није дошло до грешке.

Клијентска апликација на статус овог типа не мора да реагује, или га може приказати свом кориснику као обавештење.

Статус типа КРИТИЧАН

Овај статус значи да је приликом извршења АПИ функције дошло до критичне грешке и њеног прекида. Функција није извршена до краја, није извршила очекивану обраду и није вратила очекивани резултат.

Клијентска апликација на овакав статус реагује тако што се понаша као да је дошло до критичне грешке. Она не сме наставити рад као да је функција исправно извршена. Уколико уме, треба да на основу конкретног статуса грешке реагује на одговарајући начин. Клијентска апликација не сме да понови позив АПИ серверу са истим подацима ,осим уколико статус грешке не назначава да то треба учинити.

ШТА СВЕ ЧИНИ ОПИС ГРЕШКЕ

Када настане грешка клијентској апликацији ће бити прослеђен статус грешке. Опис грешке се уписује у *TAPIResultClass* а састоји од три податка уписана у својства *error_status*, *error_messageu error_data*:

error_status (char)

Ово је идентификатор статуса грешке. Вредност OK значи да није било грешке приликом извршавања функције. Било која друга вредност значи да је дошло до грешке.

За предвиђене грешке уводе се идентификатори статуса грешака и они су документовани у опису АПИ функције.

На основу идентификатора статуса грешке клијентска апликација може да реагује на одговарајући начин, уколико грешку познаје. Ако клијентска апликација не препознаје грешку треба да на њу реагује као да се ради о критичној грешци.

Идентификатор грешке треба проверавати као *Case Insensitive* низ знакова, односно не треба правити разлику између малих и великих слова.

error_message (char)

Текстуални опис статуса грешке. Садржи слободан текст који детаљније описује статус грешке, а који се може приказати кориснику клијентске апликације. Овај опис по правилу садржи довољно детаља да се одреди где је и због чега настала грешка.

error_data (obj)

Уколико је поред идентификатора, потребно клијенту послати и додатне информације о статусу грешке да би он могао да исправно реагује на грешку, користи се ово својство.

Ово својство може садржати скаларну вредност (на пример број документа приликом чије обраде је дошло до грешке) али и неку сложену структуру.

Ако декларација АПИ функције не садржи опис овог својства онда се оно не користи и биће празно.

ЛИСТА ОПИСА СТАТУСА ГРЕШАКА

Приликом извршавања неких функција може затребати да се клијенту врати не један, него више описа статуса грешака.

Чести случајеви:

- Пре извршења, функција проверава улазне параметре и код више параметара наилази на неисправности. Уместо да прекине извршавање и врати статус грешке на првом провереном параметру, она може да врати статусе грешака за све параметре појединачно, али одједном, као листу.

- Ако функција врши серијску обраду докумената могућа је ситуација да код обраде неких од докумената може доћи до грешке, али да функција треба да заврши обраду оних докумената где се грешка не јавља. У том случају функција може да врати листу статуса грешака за сваки обрађени документ појединачно.

ИнфоАПИ омогућава и овакав начин враћања статуса грешака. Уместо да се у *TAPIResultClass* статус грешке описује преко својстава error_status, error_message и error_data, опис статуса се може вратити клијенту као листа.

Листа статуса је листа објеката који садрже својства error_status, error_message и error_data. Са вредностима аналогним својствима у класи *TAPIResultClass*.

У листу статуса грешака се могу уписивати сви статуси грешака, укључујући и статус *OK*. Уколико сви статуси грешака у листи имају идентификатор статуса OK, онда ће и својство *TAPIResultClass.error_status* такође имати вредност *OK*.

Уколико макар један идентификатор статуса грешке у листи није OK, *TAPIResultClass.error_status* ће имати вредност *COMPLEX_ERROR_STATE*, који упућује да се статуси грешака налазе у листи. Листа статуса са грешкама се налази у *TAPIResultClass.error_list*.

Ако листа статуса грешака садржи само један статус, онда ће и основни статус грешке садржавати исте податке као тај један статус у листи.

СИСТЕМСКИ СТАТУСИ ГРЕШАКА

Системски статуси грешака су статуси који настају на системском нивоу и нису специфични за појединачне АПИ функције.

Статуси грешака специфични за АПИ функције су посебно документовани у опису сваке функције појединачно.

Системски статуси грешака на TAPIResponseEnvelope

ИДЕНТИФИКАТОР	ТИП ГРЕШКЕ	ОПИС СТАТУСА
OK	ОБАВЕШТЕЊЕ	Нема грешака
API_FUNCTION_ERROR	КРИТИЧНО	Извршена функција је вратила грешку. Детаљи о тој грешци се налазе у структури резултата функције и треба тамо проверити о којој грешци се ради.
API_PAGING_NOT_EXECUTED	КРИТИЧНО	Клијентска апликација је тражила податке са страничењем, а позвана АПИ функција не омогућава страничење
INVALID_YEAR	КРИТИЧНО	Не постоји пословна година
CARDINAL	КРИТИЧНО	Системска грешка настала у току извршавања АПИ функције
UNDEFINED	КРИТИЧНО	Дошло је до системске грешке а да АПИ функциа није ни покренута
APPLICATION_ERROR	КРИТИЧНО	Грешка настала у иницијализацији АПИ сервиса
NO_USER_RIGHTS	КРИТИЧНО	Пријављени корисник нема довољна корисничка права да покрене тражену АПИ функцију
UNKNOWN_FUNCTION	КРИТИЧНО	Не постоји позвана АПИ функција
INVALID_SERVER_LICENCE	КРИТИЧНО	Сервер није лиценциран да користи тражену АПИ функцију
INVALID_CLIENT_LICENCE	КРИТИЧНО	Клијент није лиценциран на серверу да користи тражену АПИ функцију

ИДЕНТИФИКАТОР	ТИП ГРЕШКЕ	ОПИС СТАТУСА
INVALID_FUNCTION_LICENCE	КРИТИЧНО	Серверу није дозвољено коришћење тражене АПИ функције
INVALID_MODULE_LICENCE	КРИТИЧНО	Серверу није дозвољено коришћење модула
CHECKSUM_REQUIRED	КРИТИЧНО	Порука не садржи контролни код
MODULE_LICENCE_EXPIRED	КРИТИЧНО	Истекла је лиценца за модул
CLIENT_LICENCE_EXPIRED	КРИТИЧНО	Истекла је лиценца клијента
CLIENT_NOT_LICENCED	КРИТИЧНО	Сервер нема лиценцу за овог клијента
NO_FUNCTION_CALL	КРИТИЧНО	Упит не садржи ознаку АПИ функције. Обично то значи да упит није послат POST методом.
ERROR_NO_PROCESS_DESC	КРИТИЧНО	АПИ фунција је покушала да региструје процес без описа.
ERROR_DUPLICATE_PROCESS	КРИТИЧНО	Клијент је покушао да истовремно позове исту АПИ функцију са истим улазним параметрима а она то не дозвољава.

Системски статуси грешака на TAPIResult

ИДЕНТИФИКАТОР	ТИП ГРЕШКЕ	ОПИС СТАТУСА
OK	ОБАВЕШТЕЊЕ	Нема грешака
COMPLEX_ERROR	КРИТИЧНО	АПИ функција је вратила листу грешака. Треба погледати у листу да би се утврдило како реаговати на сваку грешку појединачно.
ERR	КРИТИЧНО	Дошло је до грешке у извршавању АПИ функције. ЗАСТАРЕЛО. Не користи се.
MISSING_TABLE	КРИТИЧНО	Недостаје табела кој би требало да буде послата као параметар АПИ функције. У опису грешке може бити наведено која је то табела.
MISSING_PARAMETER	КРИТИЧНО	Недостаје параметар АПИ функције који ј еобавезан. У опису грешке може бити наведено који је то параметар.

КОНТРОЛНИ КОД

У верзији 1 је уведена обавезна употреба контролног кода приликом слања упита ИнфоАПИ-ју. Овде је објашњено како се он примењује. Ово је највећа разлика између верзија 0 и 1.

Намена контролног кода је да се провери целовитост послате поруке, обезбеди интегритет података и изврши аутентификација клијентске апликације.

Контролни код се рачуна тако што се саберу као стрингови вредности својстава *TAPIQueryEnvelope.data* и *TAPIQueryEnvelope.req_id* и и приватни кључ (добијате га уз лиценцу) и над тако добијеним стрингом израчуна се МД5 хеш.

Израчунати хеш се као хексадекадни број уписује у *TAPIQueryEnvelope.checksum.*

```
TAPIQueryEnvelope.checksum = MD5 (TAPIQueryEnvelope.data + TAPIQueryEnvelope.req_id + priv_key)
```

Ово треба радити као последњу операцију пре претварања објекта у *JSON* формат.

Провера контролног кода се ради тако то се уради исти поступак рачунања контролног кода, само се добијена вредност пореди са оним који је већ уписан у својство *TAPIResponseEnvelope.checksum* пристигле поруке.

На исти начин се рачуна контролни код и упита и резултата упита, односно као што контролни број упита проверава АПИ сервер тако клијентска апликација треба да, из безбедносних разлога, проверава контролни код одговора који добије до сервера.

Приватни кључ је низ знакова који се добија уз лиценцу а везан је за идетификатор апликације. Приватни кључ се не размењује. Њега имају и ИнфоАПИ сервер и клијентска апликација и само га употребљавају за рачунање контролног кода. ИнфоАПИ сервер према идентификатору апликације, који обавезно треба слати у свакој поруци, зна који приватни кључ треба да употреби.

ИнфоАПИ сервер неће прихватити упит који не садржи исправан контролни код који потврђује да је клијентска апликација лиценцирана.

Док развијате апликацију, можете да употребите ДЕМО лиценцу коју ћемо вам доставити на захтев. Она садржи CLIENT_ID и приватни кључ са којима можете да обављате тестирање.

Пример PHP функције која рачуна контролни код:

```
function GetChecksum($pData, $pRequestId, $pPrivateKey)  {

      $lCheckSumData = json_encode ($pDdata, JSON_UNESCAPED_UNICODE)
                        . $pRequestId
                        . $pPrivateKey;

      $lResult = strtoupper (MD5 ($lCheckSumData));

     return ($lResult);

}
```

Пример C# функције која рачуна контролни код:

```
public string GetChecksum (string pData, string pRequestId, string pPrivateKey) {

  MD5 md5 = MD5.Create();

  byte[] inputBytes = Encoding.UTF8.GetBytes(pData + pRequestId + pPrivatekey);
  byte[] hash = md5.ComputeHash(inputBytes);

  string lResult = BitConverter.ToString(hash).Replace("-", "").ToUpper();

  return lResult;

}
```

ЛИЦЕНЦИРАЊЕ

ЛИЦЕНЦА КЛИЈЕНТСКЕ АПЛИКАЦИЈЕ

У верзији 1 је уведено обавезно лиценцирање клијентске апликације. Овде је то објашњено.

Да би клијентски програм могао да позива АПИ функције потребно је да има одговарајућа овлашћења. Ова овлашћења се дефинишу лиценцом апликације која одређује коју функционалност АПИ-ја наведена апликација може да користи.

Лиценцу издаје Инфосус произвођачу клијентског програма. Ако произвођач има више програма који треба да комуницирају са АПИ-јем лиценца се издаје за сваки програм посебно.

Лиценца одређује идентификатор и приватни кључ програма. Приликом припреме упита, програм треба да у својство *TAPIQuery.client_id* упише идентификатор из лиценце и да у *TAPIQuery.checksum* упише контролни код упита који се рачуна у функцији од приватног кључа који је добијен лиценцом (погледати одељак који објашњава рачунање контролног кода).

Да би АПИ сервер прихватио упит лиценциране клијентске апликације за извршење функције морају бити испуњени следећи услови:

1. Клијентска апликација мора имати лиценцу Инфосиса да може да комуницира са АПИ-јем и да може да покрене тражену АПИ функцију.

2. АПИ сервер на који се клијентска апликација повезује мора имати лиценцу која му дозвољава да прихвата упите од дате лиценциране клијентске апликације за позиве наведеној АПИ функцији.

4. Упит мора садржавати ИД клијента и исправан контролни број израчунат уз помоћ приватног кључа из лиценце клијентске апликације.

ЛИЦЕНЦА АПИ СЕРВЕРА

У верзији 1 је уведено обавезно лиценцирање АПИ сервера. Овде је то објашњено.

АПИ сервер мора имати одговарајућу лиценцу да би могао да прихвата упите за извршавање АПИ апликација. Ову лиценцу набавља корисник Инфосис програма у складу са својим потребама.

Лиценца сервера одређује:

- које АПИ функције се могу извршавати на серверу;

- које клијентске апликације могу да се повезују са сервером и које функције да извршавају;

- укупан број корисника и број истовремених корисника на серверу;

- учесталост и број упита ка појединачним АПИ функцијама, и

- друге параметре сервера и комуникације са клијентима.