GATE Release 3.5

Ръководство за използване на интерфейса VALUES API при разработване на приложения за участниците на борсовия пазар

Том първи – Интерфейс за повикване

Съдържание

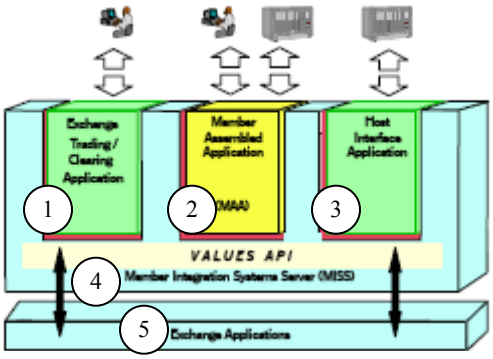
1 Въведение

Приложният програмен интерфейс VALUES API (от Virtual Access Link Using Exchange Services Application Programming Interface – Връзка за виртуален достъп до Борсови услуги с използване на API) осигурява функционалността и гъвкавостта, изисквани от стандартния отворен интерфейс към Борсовите услуги.

Спесификацията на VALUES API представлява ръководство на разработчика, предназначено за използване при дизайна и реализацията на приложения, използващи стандартния отворен интерфейс VALUES API за връзка с Борсовите услуги. Документът "Ръководство за използване на интерфейса VALUES API при разработване на приложения за участниците на борсовия пазар" представлява последната версия на спесификацията на VALUES API и описва структурата и начините за използване на интерфейса, както и съответните изисквания към развойната среда.

1.1 VALUES API

VALUES API представлява единна точка, през която системите на участниците могат да осъществяват достъп до Борсовите услуги. VALUES API може да се използва за търговски дейности, управлявани както от човек, така и от компютър. На *фигура 1.1* е показано мястото на VALUES API в общата борсова инфраструктура.



Фигура 1.1: VALUES API в контекста на борсовата инфраструктура

Легенда:
1.....Приложение за борсова търговия/клиринг
2.....Приложение, разработено от участника на борсовия пазар
3.....Приложение за връзка с приемната (хостваща) система
4.....Интеграционен сървър на участника
5.....Борсови приложения

Във *фигура 1.1* и навсякъде в спесификацията на интерфейса, "Борсово приложение" се отнася до всяко приложение, което е осигурено от Борсата и служи за предоставяне на услуги за самата търговия (например подаване на поръчки или заявки за списъци с новини), услуги за подпомагане на търговията и услуги за клиринг, предназначени за приложението на крайния потребител. "Приложение на краен потребител" (или

"потребителско приложение") се отнася за всяко приложение, което получава услуги чрез VALUES API. Освен това, в спесификацията на този интерфейс терминът "потребител" се дефинира като всеки един член, търговец или участник, който ползва Борсови услуги чрез VALUES API.

VALUES API се състои от набор технически входни точки, наричан "интерфейс за повикване" (VALUES API Call Interface) и точки за функционален достъп, наричани "приложни заявки" (VALUES API application requests):

- **Интерфейс за повикване**
  VALUES API Call Interface представлява набор от C-функции, които приложението извиква (активира), за да реализира достъп до едни или други Борсови услуги.

- **Приложни заявки**
  VALUES API application requests предоставят информация, свързана с функционалните конфигурации и формати, необходими за достъп до Борсовите услуги. Тези заявки се използват заедно с входните точки от интерфейса за повикване, представени на *фигура 2.1.*

Разделянето на техническите от функционалните компоненти във VALUES API минимизира влиянията, свързани с разработването на нови приложни заявки. На разработчиците се препоръчва да поддържат ясно разграничение между техническата рамка за обработка на повикванията към VALUES API Call Interface и функционалните аспекти като форматиране на съобщения и др.

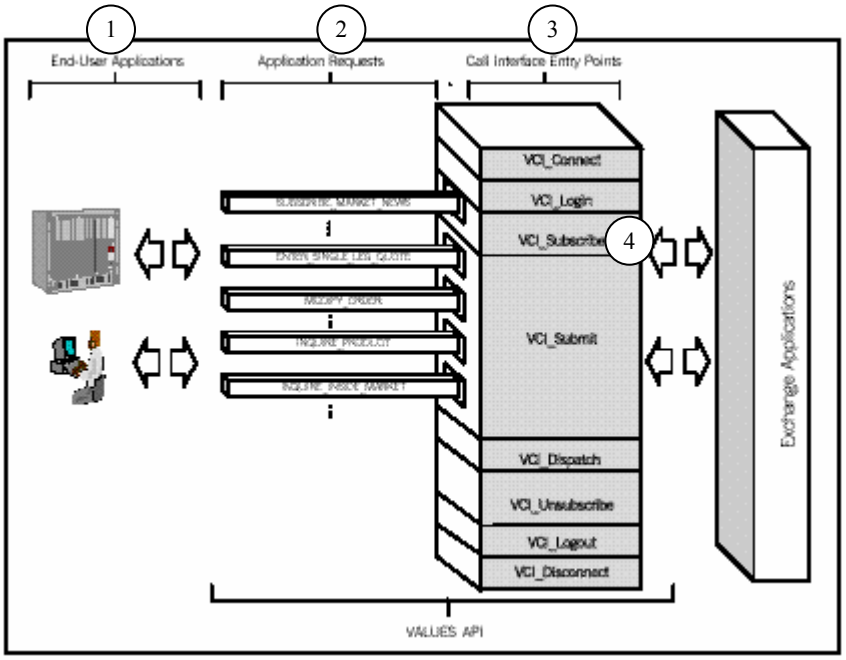
VALUES API може да бъде разширяван така, че да поддържа новата функционалност, предлагана с всяка нова версия на Борсовите приложения. Разширяването на VALUES API се постига чрез въвеждане на нови приложни заявки, съответстващи на новата функционалност във версията. Целта е да се осигури стабилност на интерфейса за повикване при последователното излизане на нови версии на VALUES API и Борсовите приложения. Освен това, в концептуално отношение VALUES API осигурява постигането на обратна съвместимост със следващите версии, които ще излизат в бъдеще.

2 VALUES API. Интерфейс за повикване. Основни понятия

В този раздел са обяснени основните понятия във VALUES API Call Interface (VCI). След кратък обзор, отделните понятия от интерфейса за повикване са обяснени по-подробно в следващите точки, където са групирани по видове услуги.

2.1 Обзор

VALUES API Call Interface представлява набор от C-функции, използвани за осъществяване на комуникация с Борсовите приложения. Чрез този интерфейс за повикване, потребителските приложения ползват Борсовите услуги, като предават приложни заявки и получават съответните приложни отговори. Освен това, чрез VALUES API Call Interface се получават и излъчвания на данни (broadcasts). Това става в отговор на заявки за абониране.



Фигура 2.1: Рамка на VALUES API – входни точки на интерфейса за повикване

Легенда:
1.....Приложения на крайния потребител
2.....Приложни заявки
3.....Входни точки на интерфейса за повикване
4.....Борсови приложения

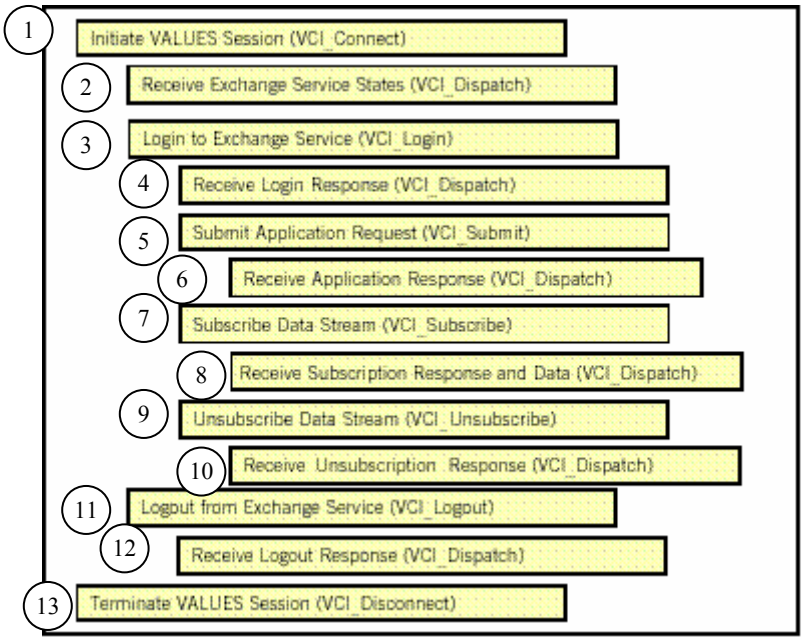
Интерфейсът за повикване се състои от фиксиран брой входни точки, показани на *фигура 2.1*, които се използват за започване на сесия, осъществяване на достъп до конкретно Борсово приложение (влизане в Борсово приложение), предаване на приложни заявки, заявяване на излъчвания (broadcasts) и получаване на отговори. На следната таблица са обобщени всички входни точки от интерфейса за повикване и е посочено групирането им по видове услуги.

Вид услуга	**Входна точка**	**Предназначение**
Управление на сесия	VCI_Connect	Използва се за установяване на комуникация между потребителското приложение и GATE (от Generic Access To Exchanges – Единен достъп до борсите)
	VCI_Disconnect	Използва се за прекратяване на достъпа до GATE
	VCI_Dispatch	Използва се за предаване на информация за първоначалния статус, специфични данни за услугите на Борсовите приложения, данни за прехода между състоянията и данни за изключения до съответната функция за обратно повикване на приложението – connect callback[(1)].
Управление на сигурността	VCI_Login	Използва се за получаване на достъп до конкретни Борсови приложения
	VCI_Logout	Използва се за прекратяване на достъпа до конкретни Борсови приложения
	VCI_Dispatch	Използва се предаване на известия и изключения във връзка с влизането/излизането до съответната функция за обратно повикване на приложението – login callback.
Управление на абонаментите	VCI_Subscribe	Използва се за искане на достъп до излъчваните потоци от данни
	VCI_Unsubscribe	Използва се за прекратяване на достъпа до излъчваните потоци от данни
	VCI_Dispatch	Използва се предаване на отговори, данни по абонамент и изключения до съответната функция за обратно повикване на приложението – subscribe callback.
Управление на заявките	VCI_Submit	Използва се за подаване на заявки към Борсовите приложения
	VCI_Dispatch	Използва се за предаване на приложни отговори и изключения до съответната функция за обратно повикване на приложението – submit callback.

[1] "Ответно повикване" (callback) е понятие, използвано при управляваното от събития програмиране (например програмирането на графичен потребителски интерфейс – GUI). С това понятие се означава специфична функция, която се изпълнява при настъпването на определено събитие.

Таблица 2.1: Входни точки на VALUES API

При използване на VALUES API Call Interface трябва да се спазва определена йерархична структура. В графичен вид, йерархията на използване е показана на *фигура 2.2*:



Фигура 2.2: Йерархия при използването на VALUES API

Легенда:
1.....Иницииране на сесия във VALUES
2.....Получаване на данни за състоянията на съответната Борсова услуга (VCI_Dispatch)
3.....Подаване на заявка за влизане в съответната Борсова услуга (VCI_Login)
4.....Получаване на отговор на заявката за влизане (VCI_Dispatch)
5.....Подаване на приложна заявка (VCI_Submit)
6.....Получаване на приложен отговор (VCI_Dispatch)
7.....Абониране за поток от данни (VCI_Subscribe)
8.....Получаване на отговор на заявката за абониране и на самите данни (VCI_Dispatch)
9.....Подаване на заявка за прекратяване на абонамента за потока от данни (VCI_Unsubscribe)
10.....Получаване на отговор на заявката за прекратяване на абонамента (VCI_Dispatch)
11.....Подаване на заявка за излизане от Борсовата услуга (VCI_Logout)
12.....Получаване на отговор на заявката за излизане (VCI_Dispatch)
13.....Прекратяване на сесията във VALUES (VCI_Disconnect)

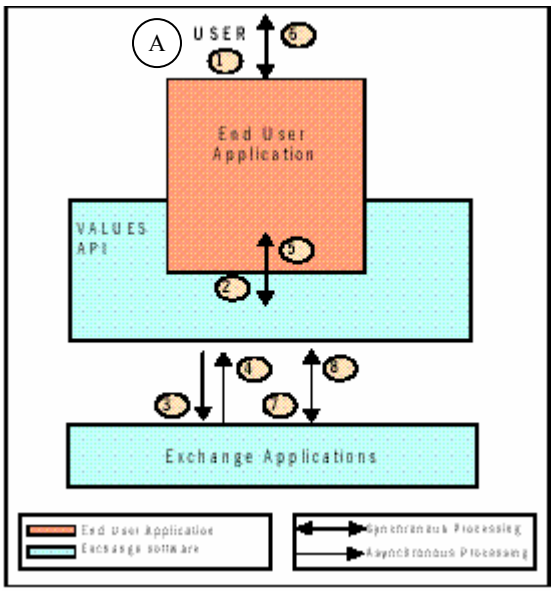
Дълбочината на отстъпа вдясно описва йерархичното ниво и показва входните точки на едно и също йерархично ниво. Намиращите се вдясно задачи/входни точки зависят то задачите/входните точки, които са вляво от тях. Например VCI_Connect е предварително условие за повикване на VCI_Login. Освен това, входните точки се използват в последователност "отгоре надолу". Редът за използване на отделните входни точки и броят възможни използвания са описани в следващите точки.

Заб.: Фигурата отразява ситуацията, когато дадена услуга Xervice реализира Broadcast Extension (разширение за излъчване на данни). Xervices, които нямат такова разширение, разрешават абониране за излъчванията само на база сесията във VALUES, тоест без валиден контекст за осъществен достъп (влизане). За повече подробности виж *точка 2.5.2*.

Входовете на VALUES API Call Interface и съответните функции за обратни повиквания към приложенията са описани в следващите точки.

Обяснени са понятията, свързани със конкретния вид услуга, например какво е сесия и какви са основните данни за управление на сесията. Освен това, обяснени са информационните потоци за всяка входна точка на интерфейса за повикване и съответните функции за обратни повиквания към приложенията при отделните видове услуги.

Информационният поток за всеки вид услуга е обяснен чрез проследяване на обработките. При обясняване на понятията във VALUES API се приема, че в модела на процесите присъства потребител, който взаимодейства с графичен потребителски интерфейс (GUI). Обработките се проследяват графично с помощта на следния модел:



Фигура 2.3: Модел за представяне на информационните потоци и обработките във VALUES API

Легенда:
A ..... Потребител
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Отделните обработки са проследени независимо от мястото, където се намира потребителското приложение – на работна станция или на интеграционния сървър на съответния участник на борсовия пазар (MISS). Тоест даден процес от потребителското приложение може да се изпълнява върху работна станция или MISS. При нормална работа се препоръчва приложенията под VALUES да се изпълняват върху работни станции. На всяка стъпка от обработката е присвоен номер, който препраща към обяснителен текст в отделна таблица. Ако дадена стъпка от обработката е поставена до главата на стрелка, това показва посоката на стъпката. Ако дадена стъпка от обработката е поставена до стеблото на стрелка, това означава и двете посоки. Таблицата с обяснителни текстове за отделните стъпки включва колони, в които са обяснени най-важните входни и изходни данни.

Обработките са описани като синхронни (двупосочни стрелки) или асинхронни (еднопосочни стрелки), гледани от страната на потребителското приложение. Кодовете/обработките в потребителските и Борсовите приложения са показани с различни нюанси на сивия цвят.

2.2 Услуги за управление на сесия

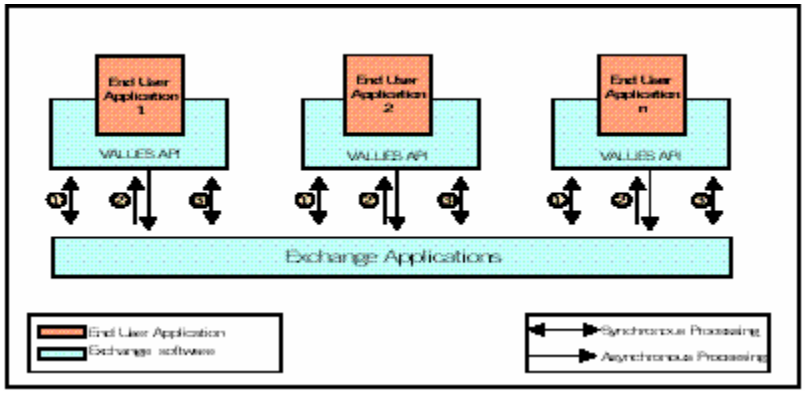
В този раздел е обяснено накратко какво представлява сесията и за какво се използват сесиите. Подробно е обяснено започването и приключването на сесия. В последната точка е представен механизмът за интегриране на приложните заявки и отговори от VCI с потребителските приложения.

2.2.1. Обзор

Сесията във VALUES е техника за контрол и управление на комуникациите между потребителското приложение и VALUES API. Всички комуникации между потребителските и Борсовите приложения през GATE се надграждат върху сесия във VALUES.

За да може да използва Борсовите услуги, потребителското приложение на първо място трябва да стартира сесия във VALUES. След като завърши взаимодействията с VALUES API, потребителското приложение трябва да приключи сесията във VALUES. Няколко потребителски приложения или няколко инстанции на едно приложение могат да се изпълняват паралелно, всяко със своя сесия във VALUES. Обаче за един приложен процес може да се открива само една сесия във VALUES.

Бележка: Интерфейсът VALUES API не е поддържа паралелно изпълнение на различни потоци (последователности от инструкции, threads). Ако многопоточно приложение излъчи повикване към входна точка на VCI в момент, когато VCI все още работи по друга последователност от инструкции, второто повикване ще е неуспешно. Освен това, входните точки на VCI не могат да се използват за многократно влизане – когато VALUES извика функция за обратно повикване, от тази функция не могат да излизат VCI повиквания. Извикванията, които нарушават това правило, се отхвърлят. Многопоточните приложения отговарят за правилното синхронизиране на своите последователности от инструкции.



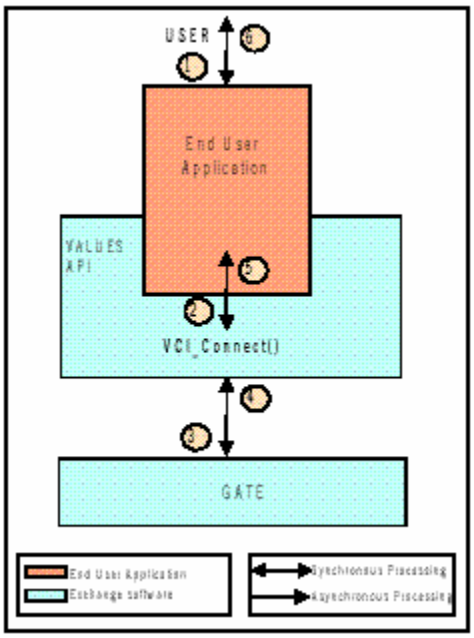
Фигура 2.4: Концептуално представяне на сесия във VALUES

Легенда:
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

На горната *фигура 2.4* е показано паралелно изпълнение на няколко потребителски приложения. Всяко приложение открива своя сесия във VALUES (1), взаимодейства с Борсовите приложения (2) и след това закрива сесията (3).

2.2.2. Откриване на сесия във VALUES

В тази точка е описана последователността от процеси при стартиране на сесия. Входната точка VCI_Connect е обяснена в контекста на потребител, който стартира потребителско приложение. VCI_Connect е предварително условие за използването на останалите входни точки от VALUES API. Процесите при откриване на сесия във VALUES са представени графично на *фигура 2.5.* Отделните стъпки са представени на *таблица 2.2*.



Фигура 2.5: Свързване с GATE

Легенда:
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Потребителят стартира потребителското приложение. Това приложение се самоинициализира	Няма	Няма
2	Приложението повиква VCI_Connect, за да започне сесия във VALUES. Предават се името на потребителя (User ID – UID), паролата (password – PW) и референция към функцията за обратно повикване. Автентичността на потребителското име и на паролата се проверява чрез функционалността за сигурност на операционната система. Референцията към функцията за обратно повикване (за тези функции виж *раздел 2.6*) се използва за предаване към приложението на информация относно наличните услуги (виж *раздел 2.9)*, началния статус и преходите между състоянията на Борсовите приложения и GATE при вече започната сесия. Името на потребителя и паролата дават възможност	UID, PW, референция към функцията за обратно повикване	Няма

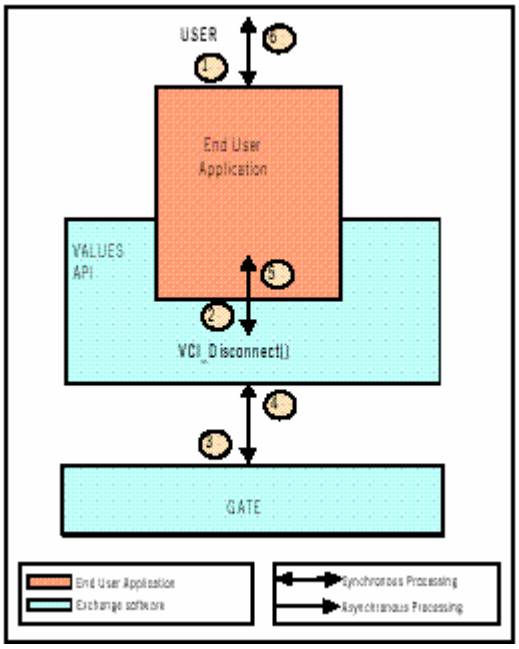
Стъпка	Описание	Вход	Изход
	за идентификация и оторизация на заявката за сесия. Приложението изчаква VCI_Connect да изпълни тази задача.		
3	VCI_Connect препраща заявката за стартиране на сесия в GATE и изчаква приключването на обработката ѝ.	UID, PW	Няма
4	GATE проверява автентичността на заявката за свързване и задава идентификатор на връзката (connection ID – CID). GATE връща на VCI_Connect код за вида изпълнение (статус) и ако заявката е успешна – съответния CID.	Няма	Статус, CID
5	VCI_Connect инсталира функция за обратно повикване на приложението, която ще бъде използвана по-късно. След това връща на потребителското приложение код за вида изпълнение (статус), експлоатационния режим (Prod Mode) и име на опашката за съобщения във VALUES (VALUES Message Queue – VMQ). Подробности за VMQ са дадени в *раздел 2.6*.	Няма	Статус, Prod Mode, име на VMQ
6	Потребителското приложение приключва собствената си инициализация, включително установяването на достъп до VMQ, след което връща контрола на потребителя. Бележка: VMQ трябва да бъде да наблюдавана за събития непосредствено след успешното започване на сесията.	Няма	Няма

Таблица 2.2: Последователност от процеси при използване на VCI_Connect

Сега потребителското приложение е установило сесия с GATE. В този момент могат да бъдат инициирани една или няколко заявки за влизане или за абониране в Борсово приложение, или сесията може да бъде прекратена.

2.2.3. Нормално прекратяване на сесия във VALUES

В тази точка е описана последователността от процеси при затваряне на сесията. Входната точка VCI_Disconnect е обяснена в контекста на потребител, който подава заявка за прекратяване на потребителското приложение. Преди да изпълни VCI_Disconnect, приложението трябва да излезе от всички Борсови приложения и да прекрати всички абонаменти. Процесите при приключване на сесията са представени графично на *фигура 2.6.* Отделните стъпки са представени на *таблица 2.3*.



Фигура 2.6: Изключване от Техническите услуги на GATE

Легенда:
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Потребителят задава команда за затваряне на потребителското приложение. Това приложение излиза от всички Борсови приложения и прекратява всички абонаменти.	Няма	Няма
2	Приложението повиква VCI_Disconnect, за да обработи заявката за прекратяване. Приложението изчаква VCI_Disconnect да изпълни тази задача.	CID	Няма

Стъпка	Описание	Вход	Изход
3	VCI_Disconnect препраща заявката за прекратяване на сесия в GATE и изчаква приключването на обработката ѝ.	CID	Няма
4	GATE проверява дали има сесия и дали потребителят има право да я прекрати. Сесията се прекратява и на VCI_Disconnect се връща код за изпълнение (статус).	Няма	Статус
5	VCI_Disconnect извиква свързаната с тази сесия функция за обратно повикване на приложението и я отменя. След това връща на приложението кода за изпълнение (статуса).	Няма	Статус
6	Сега потребителското приложение може да съобщи на потребителя за успешното изключване и да продължи последователността от процеси за затваряне.	Няма	Няма

Таблица 2.3: Последователност от процеси при използване на VCI_Disconnect

Сега потребителското приложение е приключило сесията във VALUES. В този момент приложението може да повика VCI_Connect, за да установи нова сесия с GATE.

2.2.4. Необичайно прекратяване на сесия във VALUES

При отказ на хардуерни или софтуерни компоненти могат да възникнат редица изключения. Ако приложението не може да се изключи по нормалния начин, необходимо е да се извърши обработка на изключение.

VALUES API поддържа обработката на изключения чрез статуса на изпълнение и чрез извикване на съответната функция за обратно повикване на приложението. При възникване на изключение, VALUES API реагира чрез извикване на регистрираните обратни повиквания в определена последователност. За повече информация относно обработката на изключения виж *точка 3.2.3*.

2.3 Услуги за управление на сигурността

В този раздел са разгледани в общ план услугите по управление на сигурността във VALUES API. Обяснени са входните точки VCI_Login и VCI_Logout, както и връзката им с VCI_Connect.

2.3.1. Обзор

VALUES API обезпечава две нива за сигурност на приложенията – достъп до GATE и достъп до Борсовите приложения. Обезпечаването на достъпа до GATE и Борсовите приложения се изразява в установяване на автентичността на потребителя (идентификация) и оторизиране на подадените от него заявки. Механизмът и данните, с помощта на които VALUES API осигурява сигурност на достъпа, са показани в *таблица 2.4.*

Достъпът до интерфейса за повикване на VALUES API и работата с този интерфейс подлежат на идентификация и оторизация. Контролът на достъпа се осъществява чрез серия разслоени компоненти:

- Операционната система на работната станция или на интеграционния сървър MISS
- Потребителските приложения
- Борсовите приложения

Спесификацията на VALUES API е насочена основно към сигурността, изисквана от Борсовите приложения и от потребителските приложения, които използват VALUES API. В *таблица 2.4* механизмите и отговорностите за сигурност са привързани към различните слоеве, чиято сигурност трябва да бъде осигурена. За целите на *таблица 2.4* потребител е всеки член, търговец или участник, който използва Борсови услуги чрез VALUES API.

Контрол на достъпа	Отговаря	Механизъм	Необходими данни
Работна станция и MISS	Потребителят	Определя се от потребителя	Определят се от потребителя
GATE	Борсата	VCI_Connect	Име и парола на потребителя във VALUES (Име и парола на потребителя в операционната система на MISS)
Приложения, базирани на VALUES API	Потребителят	Определя се от потребителя	Определят се от потребителя
Борсови приложения	Борсата	VCI_Login	Име на потребителя и оторизационни данни за Борсовите приложения.

Таблица 2.4: Механизми за сигурност във VALUES API

Както бе посочено в *раздел 2.4*, VCI_Connect се използва за стартиране на сесия във VALUES. Освен това, VCI_Connect е механизъм, чрез който се гарантира сигурността на достъпа до GATE.

Контролът на достъпа до Борсовите приложения се поддържа от входната точка VCI_Login. Именно VCI_Login трябва да се използва за получаване на оторизация на потребителя за всяко Борсово приложение. По-долу е описано използването на входните точки VCI_Login и VCI_Logout.

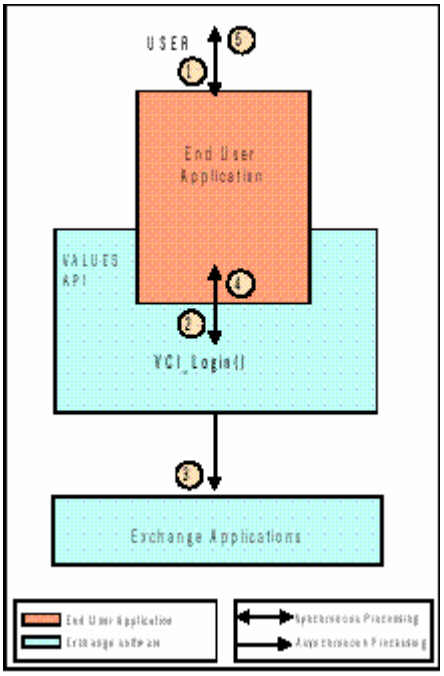
2.3.2. Влизане в Борсово приложение

В тази точка са описани процесите при обработка на заявка за влизане. Входната точка VCI_Login е описана в контекста на потребител, който опитва да получи достъп по Борсово приложение. За целта, потребителското приложение трябва да е установило сесия във VALUES чрез VCI_Connect. Една връзка с VALUES API дава възможност за множество

влизания. Подробна информация относно функционалността за поддържане на множество потребители е дадена в *раздел 2.8*.

Приложението трябва да предаде идентификатор на Xervice (dbApplID), за да посочи в кое точно Борсово приложение (=Xervice) иска да влезне. Информация за това, как се получават верни стойности на dbApplID и подробности за Xervice са дадени в *раздел 2.9*.

На *фигура 2.7* е представена в графичен вид последователността от процеси при влизане в Борсово приложение. Отделните стъпки са обяснени на *таблица 2.5*.



Фигура 2.7: Влизане в Борсово приложение

Легенда:
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Потребителското приложение иска име и оторизационни данни (auth. data) от потребителя	Няма	UID, auth. data
2	Приложението повиква VCI_Login, като предава името на потребителя, оторизационните данни (паролата), dbApplID (идентификатор на исканата	UID, auth. data, dbApplID,	Няма

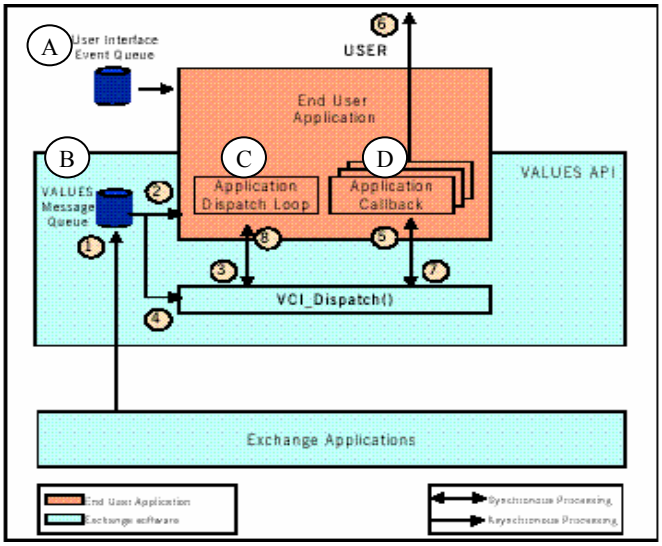
Стъпка	Описание	Вход	Изход
	Xervice) и референция към функцията за обратно повикване. Потребителското приложение изчаква VCI_Login да му върне контрола.	референция към функцията за обратно повикване	
3	VCI_Login препраща заявката за влизане на съответното Борсово приложение за обработка. VCI_Login записва референцията към функцията обратно повикване за по-късна употреба.	UID, auth. data	Няма
4	VCI_Login връща на повикващото приложение статус за предаването на искането за влизане.	Няма	Статус
5	Потребителското приложение връща контрола на крайния потребител.	Няма	Статус

Таблица 2.5: Последователност от процеси при използване на VCI_Login

Заявката за влизане е предадена на съответното Борсово приложение и е заложена функция за обратно повикване. Точният статус на обработката на заявката е неизвестен до получаване на отговора.

2.3.3. Получаване на отговор на заявката за влизане

В тази точка е описана последователността от процеси при получаване на отговор на заявка за влизане. Входната точка VCI_Dispatch и съответната функция за обратно повикване на приложението са обяснени в контекста, създаден от заявката за влизане (виж *точка 2.3.2*). На *фигура 2.8* е представена в графичен вид последователността от процеси при получаване на отговор на заявката за влизане. Отделните стъпки са обяснени на *таблица 2.6*.



Фигура 2.8: Обработка на отговор на заявка за влизане

Легенда:
A ..... Опашка от събития на потребителския интерфейс
B ..... Опашка от съобщения във VALUES
C ..... Диспечерски цикъл на приложението
D ..... Инстанции на функцията за обратно повикване на приложението
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Борсовото приложение изпраща отговор на заявката за влизане до опашката за съобщения във VALUES (VMQ).	Данни за отговора	Няма
2	Диспечерският цикъл на приложението получава съобщение за това, че е настъпило събитие във VMQ.	Събитие във VMQ	Няма
3	Потребителското приложение повиква VCI_Dispatch и изчаква изпълнение на повикването.	Няма	Няма
4	VCI_Dispatch прочита съобщението-отговор от VMQ.	Данни за отговора	Няма
5	VCI_Dispatch идентифицира функцията за обратно повикване, която е свързана със съобщението-отговор. VCI_Dispatch повиква функцията за обратно	Статус, login ID	Статус, login ID

Стъпка	Описание	Вход	Изход
	повикване на приложението (тази, която е регистрирана със заявката за влизане), като предава login ID. След това чака обръщане от функцията за обратно повикване.		
6	Функцията за обратно повикване на приложението разпознава, че съобщението съдържа отговор на заявката за влизане, като прочита предадените данни за статуса и показва на потребителя статуса на заявката за влизане. След това функцията за обратно повикване запомня login ID за последващо използване.	Статус, login ID	Статус
7	Функцията за обратно повикване се обръща към VCI_Dispatch.	Няма	Няма
8	VCI_Dispatch изтрива всичко, което знае за контекста на заявката за влизане. VCI_Dispatch връща контрола на диспечерския цикъл на приложението заедно със статус на обработката.	Няма	Статус

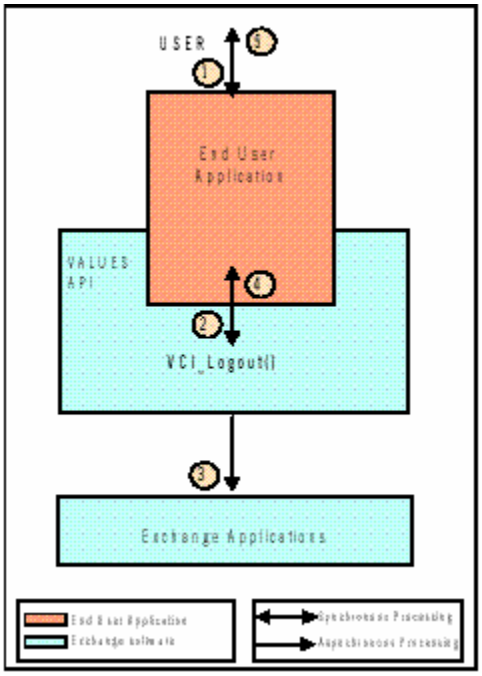
Таблица 2.6: Последователност от процеси при използване на VCI_Dispatch (получаване на отговор на заявка за влизане).

Сега потребителското приложение може да предоставя функционалност от съответното Борсово приложение (например въвеждане на поръчка).

2.3.4. Нормално излизане от Борсово приложение

В тази точка е описана последователността от процеси при обработка на заявка за излизане. Входната точка VCI_Logout е обяснена в контекста на потребител, който затваря потребителското приложение. Приема се, че по време на сесията потребителят е влязъл в Борсово приложение. Потребителското приложение може да поддържа оторизирана комуникация с Борсовото приложение през цялата сесия, като извърши необходимите действия по излизане едва при приключване на сесията. Алтернативно, услугата за излизане може да се ползва във всеки един момент по време на сесията.

На *фигура 2.9* е представена в графичен вид последователността от процеси при излизане от Борсово приложение. Отделните стъпки са обяснени на *таблица 2.7*.



Фигура 2.9: Излизане от Борсово приложение

Легенда:
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

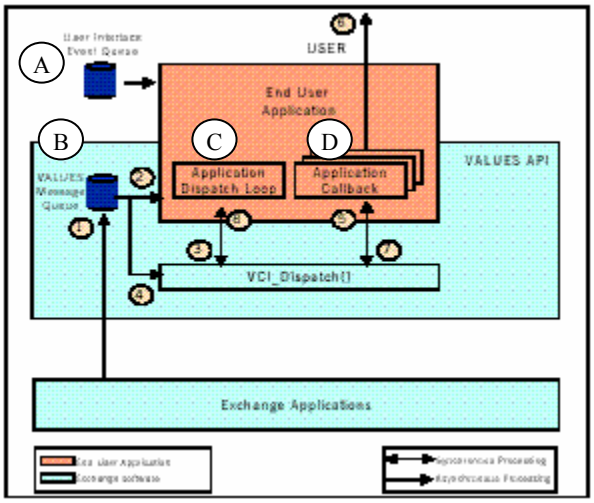
Стъпка	Описание	Вход	Изход
1	Потребителят задава команда за излизане от Борсово приложение.	Няма	Няма
2	Приложението повиква VCI_Logout, като предава dbApplID (идентификатор на Борсовото приложение) и login ID. Приложението изчаква, докато VCI_Logout изпълни тази задача.	dbApplID, login ID	Няма
3	VCI_Logout препраща заявката за излизане на съответното Борсово приложение.	UID	Няма
4	VCI_Logout връща на повикващото приложение статус за предаването на заявката за излизане.	Няма	Статус
5	Приложението връща контрола на потребителя.	Няма	Няма

Таблица 2.7: Последователност от процеси при използване на VCI_Logout

Заявката за излизане е предадена на съответното Борсово приложение. Точният статус на обработката на заявката е неизвестен до получаване на отговора.

2.3.5. Получаване на отговор на заявката за излизане

В тази точка е описана последователността от процеси при получаване на отговор на заявка за излизане. Входната точка VCI_Dispatch и съответната функция за обратно повикване на приложението са обяснени в контекста, създаден от заявката за излизане (виж *точка 2.3.4*). На *фигура 2.10* е представена в графичен вид последователността от процеси при получаване на отговор на заявката за излизане. Отделните стъпки са обяснени на *таблица 2.8*.



Фигура 2.10: Обработка на отговор на заявка за излизане

Легенда:
A ..... Опашка от събития на потребителския интерфейс
B ..... Опашка от съобщения във VALUES
C ..... Диспечерски цикъл на приложението
D ..... Инстанции на функцията за обратно повикване на приложението
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Борсовото приложение изпраща отговор на заявката за излизане до опашката за съобщения във VALUES (VMQ).	Данни за отговора	Няма
2	Диспечерският цикъл на приложението получава	Събитие във	Няма

Стъпка	Описание	Вход	Изход
	съобщение за това, че е настъпило събитие във VMQ.	VMQ	
3	Потребителското приложение повиква VCI_Dispatch и изчаква изпълнение на повикването.	Няма	Няма
4	VCI_Dispatch прочита съобщението-отговор от VMQ.	Данни за отговора	Няма
5	VCI_Dispatch идентифицира функцията за обратно повикване, която е свързана със съобщението-отговор. VCI_Dispatch повиква функцията за обратно повикване на приложението (тоест, при получаване на отговор на заявката за излизане се използва функцията за обратно повикване, която е била регистрирана със съответната заявка за влизане), като предава login ID. След това чака обръщане от функцията за обратно повикване.	Статус, login ID	Статус, login ID
6	Функцията за обратно повикване на приложението разпознава, че съобщението съдържа отговор на заявката за излизане, като прочита предадените данни за статуса и показва на потребителя статуса на заявката за излизане. Ако е приложимо, изчиства данните за Борсовото приложение, което е напуснато току-що.	Статус	Статус
7	Функцията за обратно повикване се обръща към VCI_Dispatch.	Няма	Няма
8	VCI_Dispatch изтрива всичко, което знае за контекста на заявката за излизане. VCI_Dispatch връща контрола на диспечерския цикъл на приложението заедно със статус на обработката.	Няма	Статус

Таблица 2.8: Последователност от процеси при използване на VCI_Dispatch (получаване на отговор на заявка за излизане).

Сега потребителят е излязъл успешно от Борсовото приложение.

2.3.6. Необичайно излизане от Борсово приложение

При използването на GATE могат да възникнат редица изключения, например отказ на хардуерни и/или софтуерни компоненти. Ако приложението не може да излезе по нормалния начин, необходимо е да се извърши обработка на изключение.

VALUES API поддържа обработката на изключения чрез статуса на изпълнение и чрез извикване на съответната функция за обратно повикване на приложението. При възникване на изключение, VALUES API реагира чрез извикване на регистрираните обратни повиквания в определена последователност. За повече информация относно обработката на изключения виж *точка 3.2.3*.

2.4 Услуги за управление на заявките

В този раздел е обяснено накратко какво представляват заявките и как те биват обработвани от VALUES API. Обяснена е входната точка VCI_Submit и начина, по който тази точка се използва за изпращане на заявки към Борсовите приложения.

2.4.1. Обзор

Услугите за управление на заявките осигуряват достъп до Борсовите приложения. Всяка отработена заявка се състои от една приложна заявка и един приложен отговор.

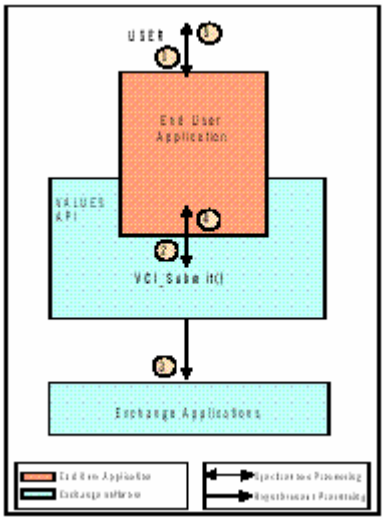
Приложната заявка се използва за задействане на една или друга обработка в Борсовото приложение. Тя се състои от идентификатор (ID) на заявката, който е уникален за всяко поддържано Борсово приложение, както и всички данни, необходими за изпълнение на заявката (например данни за поръчката, необходими за въвеждане на поръчка). Подробна информация за различните заявки е дадена в томовете за съответните Борсови приложения.

Приложният отговор съдържа резултатите от обработката, извършена от Борсовото приложение. Отговорът се генерира от сезираното Борсово приложение и се предава асинхронно към потребителското приложение. Приложният отговор се състои от идентификатора (ID) на съответната заявка, статус за изпълнение на обработката и допълнителни данни, необходим за съобщаване на резултата от обработката. На една приложна заявка се получава един-единствен приложен отговор.

2.4.2. Подаване на приложна заявка

В тази точка е описана последователността от процеси при подаване на приложна заявка. Входната точка VCI_Submit е обяснена в контекста на потребител, който подава поръчка. Приема се, че преди това потребителят е получил необходимите оторизации (за данни относно сигурността и влизането виж *раздел 2.3*).

Процесите при подаване на приложна заявка са представени графично на *фигура 2.11*. Отделните стъпки са описани на *таблица 2.9*, като за пример е използвана заявката "Въвеждане на единична поръчка".



Фигура 2.11: Обработка на приложна заявка

Легенда:
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Потребителят въвежда поръчка в своето приложение.	Данни за поръчката	Няма
2	Потребителското приложение форматира приложна заявка "Въвеждане на еднократна поръчка" и повиква VCI_Submit, за да я изпрати до Борсовото приложение. Потребителското приложение включва референция към функцията за обратно повикване, която ще бъде извикана след изпълнението на заявката. Потребителското приложение чака VCI_Submit да му върне контрола.	Данни за поръчката, референция към функцията за обратно повикване, login ID, dbApplID	Няма
3	VCI_Submit препраща приложната заявка към съответното Борсово приложение. VCI_Submit записва референцията към функцията за обратно повикване за по-късна употреба.	Данни за поръчката, dbApplID, UID	Няма
4	VCI_Submit връща на повикващото приложение статус за предаването на приложната заявка.	Няма	Статус

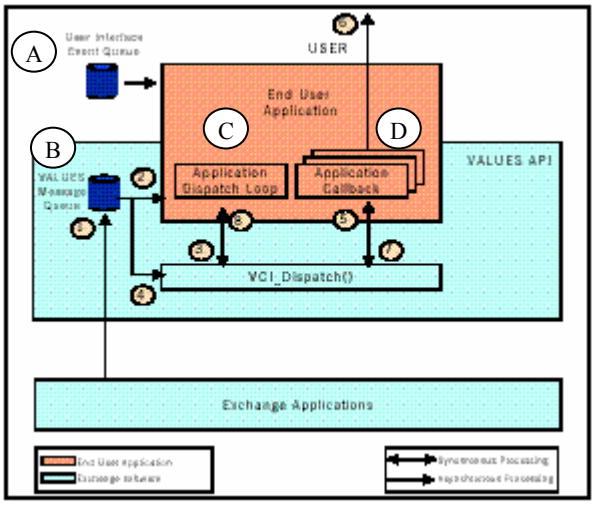
Стъпка	Описание	Вход	Изход
5	Приложението връща контрола на потребителя.	Няма	Няма

Таблица 2.9: Последователност от процеси при използване на VCI_Submit

Поръчката на потребителя е предадена на съответното Борсово приложение и е заложена функция за обратно повикване. Точният статус на обработката на заявката е неизвестен до получаване на отговора.

2.4.3. Получаване на отговор на приложна заявка (приложен отговор)

В тази точка е описана последователността от процеси при получаване на приложен отговор (т.е. отговор на приложна заявка). Входната точка VCI_Dispatch и съответната функция за обратно повикване са обяснени в контекста, създаден от подаването на поръчката (виж *точка 2.4.2*). Процесите при получаване на приложен отговор са представени графично на *фигура 2.12*. Отделните стъпки са описани на *таблица 2.10*.



Фигура 2.12: Обработка на отговор на приложна заявка (приложен отговор)

Легенда:
A ..... Опашка от събития на потребителския интерфейс
B ..... Опашка от съобщения във VALUES
C ..... Диспечерски цикъл на приложението
D ..... Инстанции на функцията за обратно повикване на приложението
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Борсовото приложение изпраща приложния отговор до опашката за съобщения във VALUES (VMQ).	Данни за отговора	Няма
2	Диспечерският цикъл на приложението получава съобщение за това, че е настъпило събитие във VMQ.	Събитие във VMQ	Няма
3	Потребителското приложение повиква VCI_Dispatch и изчаква изпълнение на повикването.	Няма	Няма
4	VCI_Dispatch прочита съобщението-отговор от VMQ.	Данни за отговора	Няма
5	VCI_Dispatch идентифицира функцията за обратно повикване, която е свързана със съобщението-отговор. VCI_Dispatch повиква функцията за обратно повикване на приложението, като предава данни за приложния отговор и данни за заявката. След това чака обръщане от функцията за обратно повикване.	Данни за отговора, данни за заявката	Няма
6	Функцията за обратно повикване на приложението показва получените данни на потребителя.	Няма	Няма
7	Функцията за обратно повикване се обръща към VCI_Dispatch.	Няма	Няма
8	VCI_Dispatch изтрива всичко, което знае за контекста на приложната заявка, приложния отговор и функцията за обратно повикване. VCI_Dispatch връща контрола на диспечерския цикъл на приложението заедно със статус на обработката.	Няма	Статус

Таблица 2.10: Последователност от процеси при използване на VCI_Dispatch (получаване на отговор на приложна заявка).

Приложението е получило асинхронния отговор и го е показало на потребителя.

2.5 Услуги за управление на абонаменти

В този раздел са разгледани в общ план механизма на абониране, потоците от данни и излъчванията (емисиите) на данни. В следващите точки са описани входните точки на интерфейса за повикване, които се използват за стартиране на абонирането, получаване на данни по абонамент и прекратяване на абонирането.

2.5.1. Обзор

Абонирането е механизъм за поискване на управлявана от събития информация, предоставяна от Борсовите приложения. Абонирането може да бъде стартирано и спирано. Поисканата абонаментна информация пристига асинхронно.

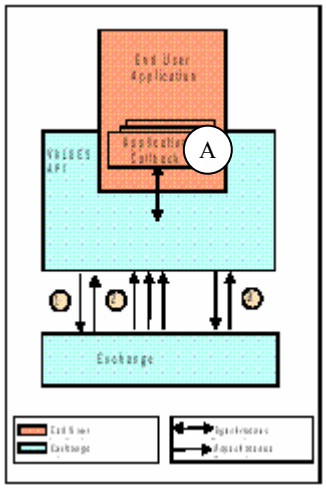
Бележка: Тъй като излъчваните данни понякога са в големи обеми, необходимо е да се напишат бързодействащи функции за обратни повиквания. За повече подробности виж *точка 3.9.1.*

Абонаменти се поддържат за потоци от данни (набори от избрани данни), чрез които се разпространява публична и частна информация от Борсовите приложения. Всяка Борса предоставя своите потоци от публични данни на всички участници на съответния пазар. Публичните пазарни данни, като "най-добри цени", "изтъргувани обеми" или "новини" се излъчват чрез публични потоци от данни. Частните потоци от данни се съставят индивидуално за участниците на Борсата и съдържат информация за елементи като "собствени поръчки" и др.

Абонирането за поток от данни е валидно за определено време през сесията във VALUES. Абонирането може да бъде задействано еднократно или многократно.

Веднага след като потребителят се е абонирал за поток от данни, излъчванията (емисиите) на данни ще се получават непрекъснато при генерирането им от Борсовите приложения.

Услугите, предоставяни от функционалността за управление на абонаменти, са обобщени на *фигура 2.13*.



Фигура 2.13: Концептуално представяне на функционалността за абониране

Легенда:
A ..... Инстанции на функцията за обратно повикване на приложението
.. Потребител
.. Борса
... Синхронна обработка
... Асинхронна обработка

Заявката за абониране съдържа данни за идентификация на желания поток от данни и се изпраща до съответното Борсово приложение, а отговорът на заявката се получава асинхронно (1). След това, всички нови данни, предназначени за излъчване, се изпращат на потребителското приложение асинхронно (2) чрез функцията за обратно повикване, посочена в заявката за абониране. Потребителското приложение може да прекрати абонамента във всеки едни момент. За целта то трябва да изпрати заявка за прекратяване на абонамента, която се потвърждава от отговор за прекратяване на абонамента (3).

2.5.2. Broadcast Extension

Започвайки от GATE версия 3.0, Борсовите приложения могат да реализират разширения за излъчване на данни (Broadcast Extensions). За да проверите дали дадена версия на Xervice реализира това разширение, молим вижте томовете от документацията на VALUES API, които се отнасят за тази Xervice.

Ако е реализирано Broadcast Extension, активни са следните характеристики:

- Разрешаване на излъчвания (Broadcast Authorization): При всяко едно абониране за излъчвания трябва да има валиден контекст за влизане, подобно на функционалността за управление на заявките. Тази характеристика е забранена за приложения, които

използват обратна съвместимост към версията на интерфейса за повикване посредством CVN_011.

- Обединяване на излъчвания (Broadcast Clustering): За някои обемисти потоци от излъчвани данни, няколко съобщения могат да бъдат обединени в едно VMQ съобщение, с което се постигат значителни икономии на служебни входно-изходни данни. Тази характеристика е налична независимо от това, кой CVN се използва от потребителското приложение. Следователно, тя не може да бъда забранена с поддържания от GATE механизъм за обратна съвместимост.

2.5.3. Идентифициране на наличните потоци от данни

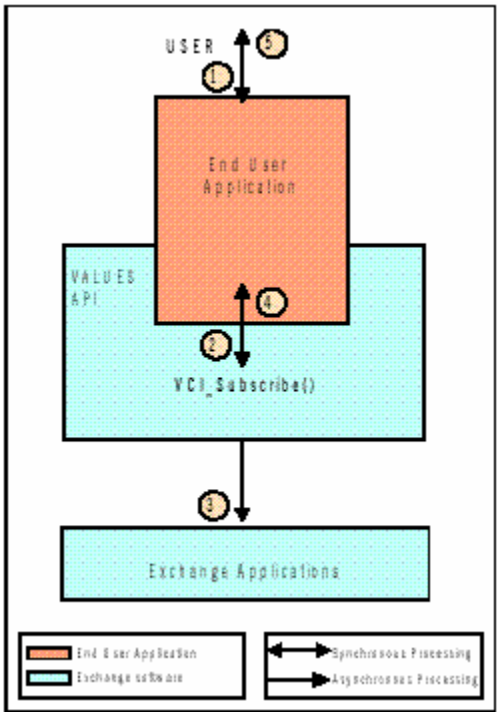
Абонирането за поток от данни се прави, като се посочи темата на абонамента (Борсова услуга, вид на потока от данни, критерии за филтриране), както и функция за обратно повикване, която да получава данните, за които се отнася абонаментът. Това се прави при повикването на VCI_Subscribe. За повече информация относно темите и наличните потоци от данни, молим отнесете се до тома от документацията, който се отнася за конкретното Борсово приложение. Интерфейсът за повикване на VALUES API предлага обща услуга за възстановяване на прекъснати излъчвания. За тази цел е предвидена специална абонаментна тема за предаване на пропуснати съобщения (Gap Notifications) за всеки поток, в който тази тема се използва. Повече подробности са дадени в *раздел 3.9*.

2.5.4. Абониране за поток от данни

В тази точка е описана последователността от процеси при абониране за потоци от данни. Входната точка VCI_Subscribe е обяснена в контекста на потребител, който отваря прозорец. Приема се, че потребителят вече е получил необходимите оторизации, ако съответната Xervice реализира Broadcast Extension (виж *точка 2.5.2*, за подробности относно сигурността и влизането виж *раздел 2.3*).

Приложението трябва да предаде идентификатор на Xervice (dbApplID), за да посочи за данни от коя точно Борсова услуга (=Xervice) желае да се абонира. Информация за това, как се получават верни стойности на dbApplID, е дадена в раздел *2.9*.

На *фигура 2.14* е представена в графичен вид последователността от процеси при абониране за потоци от данни. Отделните стъпки са обяснени на *таблица 2.11*.



Фигура 2.14: Абониране за поток от данни

Легенда:
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Потребителят отваря прозорец за показ на пазарна информация (приложението донася първоначален набор от данни, използвайки издирване – Inquiry).	Няма	Няма
2	Приложението повиква входната точка VCI_Subscribe, като предава темата (subject) на потока от данни и референция към функцията за обратно повикване. Потребителското приложение изчаква, докато VCI_Subscribe приключи тази задача.	subject, референция към функцията за обратно повикване, dbApplID	Няма
3	VCI_Subscribe препраща заявката за абониране на локалната архитектура.	subject, dbApplID	Няма

Стъпка	Описание	Вход	Изход
	VCI_Subscribe записва референцията към функцията за обратно повикване, която ще бъде използвана по-късно.		
4	VCI_Subscribe съобщава на повикващото приложение статус за предаването на заявката за абониране.	Няма	Статус
5	Приложението връща контрола на потребителя.	Няма	Няма

Таблица 2.11: Последователност от процеси при използване на VCI_Subscribe

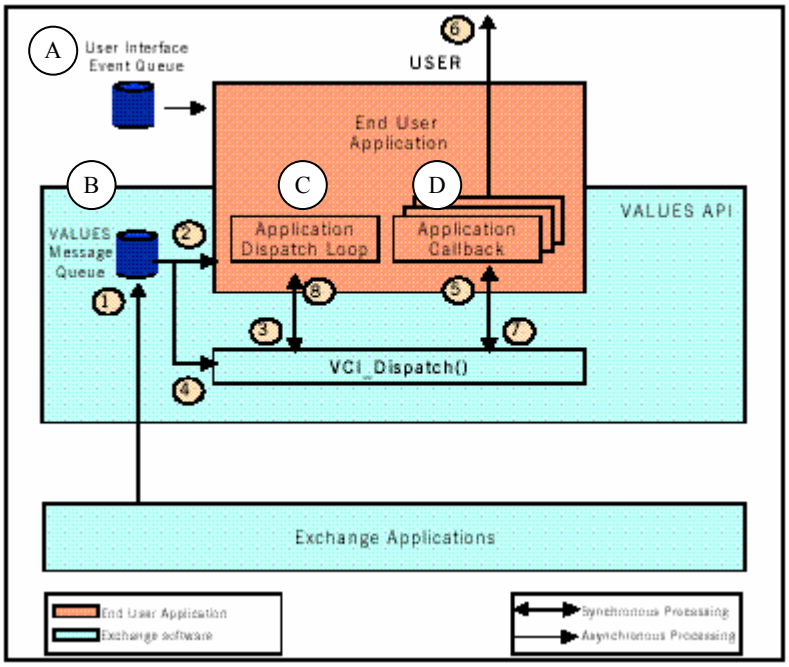
Заявката за абониране е предадена на Борсовото приложение. Точният статус на обработката на заявката за абониране остава неизвестен, докато се получи отговор на тази заявка.

Регистрираната от VCI_Subscribe функция за обратно повикване се използва за обработване на:

- (единичен) отговор на заявката за абониране (*точка 2.5.5*);
- излъчвания (емисии) на данни, за които се отнася абонаментът (*точка 2.5.6*).

2.5.5. Получаване на отговор на заявка за абониране

В тази точка е обяснена последователността от процеси, която се изпълнява при асинхронното пристигане на отговора на заявката за абониране. Този отговор показва статуса на подадената преди това заявка за абониране. Входната точка VCI_Dispatch и съответната функция за обратно повикване на приложението са описани в контекста, създаден от заявката за абониране. На *фигура 2.15* е представена в графичен вид последователността от процеси при получаване на отговора на заявката за абониране. Отделните стъпки са обяснени на *таблица 2.12*.



Фигура 2.15: Обработка на отговор на заявка за абониране

Легенда:
A ..... Опашка от събития на потребителския интерфейс
B ..... Опашка от съобщения във VALUES
C ..... Диспечерски цикъл на приложението
D ..... Инстанции на функцията за обратно повикване на приложението
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Борсовото приложение изпраща отговор на заявката за абониране до опашката за съобщения във VALUES (VMQ).	Данни за отговора	Няма
2	Диспечерският цикъл на приложението получава съобщение за това, че е настъпило събитие във VMQ.	Събитие във VMQ	Няма
3	Потребителското приложение повиква VCI_Dispatch и изчаква изпълнение на повикването.	Няма	Няма
4	VCI_Dispatch прочита съобщението-отговор от VMQ.	Данни за отговора[1]	Няма

Стъпка	Описание	Вход	Изход
5	VCI_Dispatch идентифицира функцията за обратно повикване, която е свързана със съобщението-отговор. VCI_Dispatch повиква функцията за обратно повикване на приложението, като предава subscription ID и статус. След това чака обръщане от функцията за обратно повикване.	Статус, subsID	Статус
6	Функцията за обратно повикване на приложението разпознава, че съобщението съдържа отговор на заявката за абониране, като прочита предадените данни за статуса и показва на потребителя статуса на заявката за абониране. След това записва subscription ID за по-нататъшно ползване.	Статус, subsID	Статус
7	Функцията за обратно повикване се обръща към VCI_Dispatch.	Няма	Няма
8	VCI_Dispatch връща контрола на диспечерския цикъл на приложението заедно със статус на обработката.	Няма	Статус

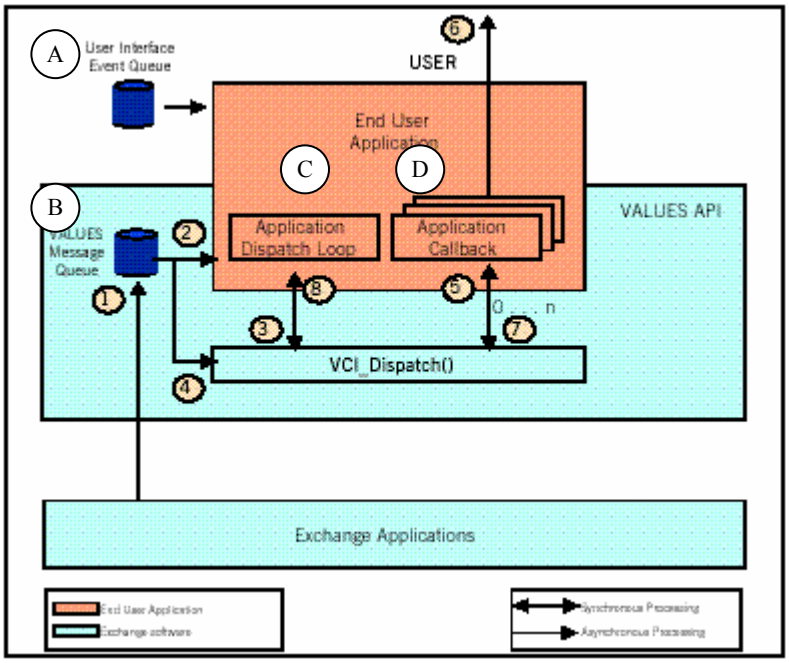
[1] Данните за отговора съдържат статуса на заявката за абониране. Тук не се включват самите излъчвания на данни.

Таблица 2.12: Последователност от процеси при използване на VCI_Dispatch (получаване на отговор на заявка за абонамент).

Потребителското приложение се абонирало за поток от данни и ще получава тези данни през една и съща регистрирана функция за обратно повикване.

2.5.6. Получаване на данни по абонамент

В тази точка е обяснена последователността от процеси, която се изпълнява при асинхронното пристигане на данни по абонамент (т.е. на излъчвания). Входната точка VCI_Dispatch и съответната функция за обратно повикване на приложението са описани в контекста, създаден от абонамента. На *фигура 2.16* е представена в графичен вид последователността от процеси при получаване на данни по абонамент. Отделните стъпки са обяснени на *таблица 2.13*.



Фигура 2.16: Получаване на данни по абонамент

Легенда:
A ..... Опашка от събития на потребителския интерфейс
B ..... Опашка от съобщения във VALUES
C ..... Диспечерски цикъл на приложението
D ..... Инстанции на функцията за обратно повикване на приложението
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Борсовото приложение изпраща абонаментните данни до опашката за съобщения във VALUES (VMQ).	Данни по абонамент	Няма
2	Диспечерският цикъл на приложението получава съобщение за това, че е настъпило събитие във VMQ.	Събитие във VMQ	Няма
3	Потребителското приложение повиква VCI_Dispatch и изчаква изпълнение на повикването.	Няма	Няма
4	VCI_Dispatch прочита съобщението-отговор от VMQ.	Данни по абонамент	Няма

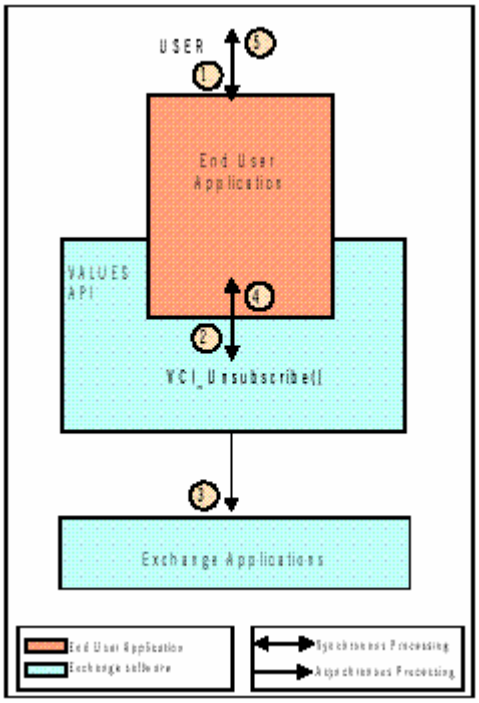
Стъпка	Описание	Вход	Изход
5	VCI_Dispatch идентифицира функцията за обратно повикване, която е свързана с абонамента. VCI_Dispatch повиква функцията за обратно повикване на приложението, като предава данните по абонамента. След това чака обръщане от функцията за обратно повикване. Възможни са няколко извиквания (активирания) на функцията за обратно повикване. При определени обстоятелства тази функция може изобщо да не бъде извикана, което е равносилно на това, че VALUES изхвърля ненужното съобщение от VMQ.	Данни по абонамент	Данни по абонамент
6	Функцията за обратно повикване на приложението разпознава, че съобщението съдържа данни по абонамент, като прочита предадените данни за статуса. Приложението показва на потребителя данните по абонамента, като попълва или актуализира прозореца, инициализиран при откриване на абонамента.	Няма	Данни по абонамент
7	Функцията за обратно повикване се обръща към VCI_Dispatch.	Няма	Няма
8	VCI_Dispatch връща контрола на диспечерския цикъл на приложението заедно със статус на обработката.	Няма	Статус

Таблица 2.13: Последователност от процеси при използване на VCI_Dispatch (получаване на данни по абонамент).

Докато абонаментът е активен, функцията за обратно повикване ще бъде викана при пристигането на нови данни. Потребителят може да прекрати абонирането по всяко време или да се абонира за други потоци от данни.

2.5.7. Нормално прекратяване на абонамента за поток от данни

В тази точка е описана последователността от процеси при прекратяване на абонирането. Входната точка VCI_Unsubscribe е описана в контекста на потребител, който затваря прозореца и по този начин задейства заявка за прекратяване на абонирането. Процесите при прекратяване на абонирането са представени графично на *фигура 2.17.* Отделните стъпки са представени на *таблица 2.14*.



Фигура 2.17: Прекратяване на абонамент за поток от данни

Легенда:
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

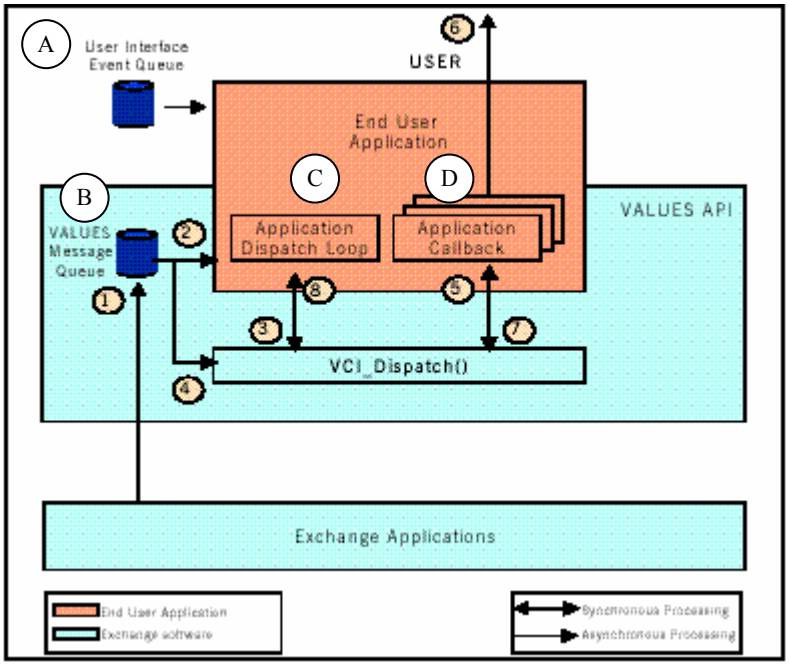
Стъпка	Описание	Вход	Изход
1	Потребителят затваря прозореца за преглед на пазара.	Няма	Няма
2	Приложението повиква входната точка VCI_Unsubscribe, като предава subscription ID. Потребителското приложение изчаква, докато VCI_Unsubscribe изпълни тази задача.	subsID	Няма
3	VCI_Unsubscribe препраща заявката за прекратяване на абонамента до локалната архитектура.	subsID	Няма
4	VCI_Unsubscribe връща на повикващото приложение статус за предаването на заявката за прекратяване.	Няма	Статус
5	Приложението връща контрола на потребителя	Няма	Няма

Таблица 2.14: Последователност от процеси при използване на VCI_Unsubscribe

Заявката за прекратяване на абонамента е предадена на съответното Борсово приложение. Точният статус на обработката на заявката остава неизвестен до получаване на отговора.

2.5.8. Получаване на отговор на заявка за прекратяване на абонамент

В тази точка е обяснена последователността от процеси, която се изпълнява при асинхронното пристигане на отговора на заявката за прекратяване на абонамент. Този отговор показва статуса на подадена по-рано заявка за прекратяване. Входната точка VCI_Dispatch и съответната функция за обратно повикване на приложението са описани в контекста, създаден от заявката за прекратяване на абонамента. На *фигура 2.18* е представена в графичен вид последователността от процеси при получаване на отговор на заявката за прекратяване на абонамента. Отделните стъпки са обяснени на *таблица 2.15*.



Фигура 2.18: Получаване на отговор на заявка за прекратяване на абонамент

Легенда:
A ..... Опашка от събития на потребителския интерфейс
B ..... Опашка от съобщения във VALUES
C ..... Диспечерски цикъл на приложението
D ..... Инстанции на функцията за обратно повикване на приложението
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	Борсовото приложение изпраща отговор на заявката за прекратяване на абонамента до опашката за съобщения във VALUES (VMQ).	Данни за отговора	Няма
2	Диспечерският цикъл на приложението получава съобщение за това, че е настъпило събитие във VMQ.	Събитие във VMQ	Няма
3	Потребителското приложение повиква VCI_Dispatch и изчаква изпълнение на повикването.	Няма	Няма
4	VCI_Dispatch прочита съобщението-отговор от VMQ.	Данни за отговора	Няма
5	VCI_Dispatch идентифицира функцията за обратно повикване, която е свързана със съответния абонамент. VCI_Dispatch извиква функцията за обратно повикване на приложението, като предава subscription ID и статус. След това чака обръщане от функцията за обратно повикване.	Статус, subsID	Статус, subsID
6	Функцията за обратно повикване на приложението разпознава, че съобщението съдържа отговор на заявката за прекратяване на абонамент, като прочита предадените данни за статуса. Приложението показва на потребителя статуса на заявката за прекратяване на абонамента. Ако е приложимо, тя изчиства данните за излъчвания поток, абонаментът за който е бил прекратен току-що.	Статус	Статус
7	Функцията за обратно повикване се обръща към VCI_Dispatch.	Няма	Няма
8	VCI_Dispatch връща контрола на диспечерския цикъл на приложението заедно със статус на обработката.	Няма	Статус

Таблица 2.15: Последователност от процеси при използване на VCI_Dispatch (получаване на отговор на заявка за прекратяване на абонамент).

Потребителското приложение е прекратило абонамента за съответния поток от данни.

2.5.9. Необичайно прекратяване на абонамента за поток от данни

При използването на GATE могат да възникнат редица изключения, например отказ на хардуерни и/или софтуерни компоненти. Ако приложението не може да прекрати абонамента по нормалния начин, необходимо е да се извърши обработка на изключение.

VALUES API поддържа обработката на изключения чрез статуса на изпълнение и чрез извикване на съответната функция за обратно повикване на приложението. При възникване на изключение, VALUES API реагира чрез извикване на регистрираните обратни повиквания в определена последователност. За повече информация относно обработката на изключения виж *точка 3.2.3*.

2.6 Интегриране на събитията от VALUES

В този раздел е описан механизмът за интегриране на събитията от VALUES в потребителското приложение. Интеграцията е базирана на получаване на събитията от VALUES в потребителското приложение и препращане на тези събития до VALUES API. Потребителското приложение отговаря за получаване на събитията от VALUES посредством опашка, създавана от VCI_Connect при стартиране на VALUES. Опашката се предоставя от операционната система и е базирана на сокет (socket) интерфейс. Опашката позволява общ достъп както от VALUES, така и от потребителското приложение. Името на опашката за съобщения (идентификатор на сокета, VMQname) се връща от VCI_Connect, за да може потребителското приложение да има достъп до нея. От приложението се изисква да наблюдава опашката за събития и да ги предава на API при настъпването им.

Във VALUES има четири вида събития:

- Отговори. Тези събития са отговори на заявките, изпращани от потребителското приложение до Борсовите приложения през входните точки VCI_Submit, VCI_Login, VCI_Logout, VCI_Subscribe и VCI_Unsubscribe.

- Излъчвания. Това са събития за получаване на асинхронни данни по абонамент за даден поток през входната точка VCI_Subscribe.

- Изключения. Представляват събития, изпращани до VALUES при проблеми с връзката (напр. неизправности по мрежата, отказ на GATE и др.).

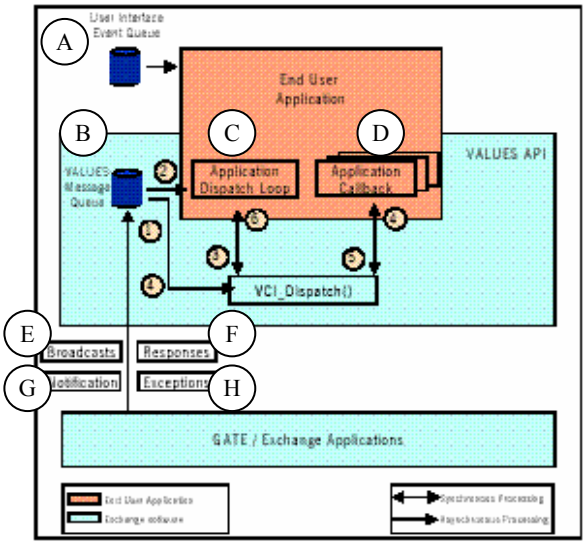
- Известия (нотификации). Това са събития, изпращани до VALUES при промяна на разполагаемостта на дадено Борсово приложение (например начална разполагаемост на Борсовото приложение след рестартирането му).

За да не се прекъсват синхронните повиквания към VALUES API (т.е. VCI_Connect, VCI_Disconnect), базираното на VALUES API приложение трябва да забранява асинхронни събития (например сигнали от UNIX).

Като общо правило, базираните на VALUES API потребителски приложения контролират процеса, т.е. основният обработващ цикъл е реализиран в потребителското приложение. VALUES API е библиотека и няма обработващ цикъл. Следователно, следенето за събития (например събития от VALUES, входни сигнали от клавиатурата или мишката) е задължение на потребителското приложение. Потребителското приложение предава (т.е. диспечира с използване на VCI_Dispatch) контрола върху обработката временно на VALUES API, за да позволи обработката на събитията от VALUES.

Функциите за обратно повикване на приложението имат специфичен функционален прототип и клиентска (индивидуална) имплементация. Функционалните прототипи на тези функции са дефинирани в спесификацията на VALUES API, а самите функции за обратно повикване се реализират от потребителското приложение. Потребителското приложение може да регистрира функции за обратни повиквания в повечето входни точки, като предаде референция към съответната функция. Когато потребителското приложение получи събитие от VALUES, то предава обработката на VALUES API. На свой ред VALUES API прочита данните за събитието от VMQ, разпознава регистрираната референция към функцията за обратно повикване, форматира свързаните със събитието данни, извиква функцията за обратно повикване и предава данните на потребителското приложение.

На *фигура 2.19* е представена в графичен вид последователността от процеси при предаване на асинхронни отговори, излъчвания (емисии) на данни и изключения. Отделните стъпки са обяснени на *таблица 2.16*.



Фигура 2.19: Предаване на съобщение от VALUES до потребителското приложение

Легенда:
A ..... Опашка от събития на потребителския интерфейс
B ..... Опашка от съобщения във VALUES
C ..... Диспечерски цикъл на приложението
D ..... Инстанции на функцията за обратно повикване на приложението
E ..... Излъчвания (емисии) на данни
F ..... Отговори
G ..... Известия
H ..... Изключения
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

Стъпка	Описание	Вход	Изход
1	При повикване на VCI_Connect, VALUES API създава опашка (VMQ), в която може да пише и друг процес. Борсовите приложения, които използват GATE, записват събитията от VALUES във VMQ.	Събитие от VALUES	Няма
2	Потребителското приложение е длъжно да проверява VMQ за събития (т.е. като "прослушва" VMQ за събития със собствения си цикъл за диспечиране на събития).	Известие за предстоящо събитие	Няма
3	Когато потребителското приложение разбере, че във VMQ е настъпило събитие, то повиква VCI_Dispatch изчаква повикването да приключи.	Няма	Няма
4	VCI_Dispatch прочита свързаното със събитието съобщение от VMQ и идентифицира функцията за обратно повикване, която трябва да извика. Възможни са неколкократни извиквания на една и съща или на различни функции за обратни повиквания. VCI_Dispatch: Предава данните за заявката или за отговора (ако се касае за отговор). Предава излъчваните данни (ако се касае за излъчване). Връща известие (първоначално състояние на Борсовото приложение или промяна на състоянието на Борсово приложение). Връща изключение (ако е настъпило изключение). Извиква висящите обратни повиквания (при прекъсване на връзка и излизане, неразполагаемост на Борсово приложение и непрозрачно прекъсване).	Съобщение от VALUES	Няма
5	Функцията за обратно повикване на приложението определя вида на полученото съобщение (т.е. отговор, излъчване, известие или изключение), като прочита предадените данни за статуса. След това функцията за обратно повикване показва получената информация на екран и се обръща към VCI_Dispatch.	Статус	Получената информация
6	VCI_Dispatch връща информация за статус на цикъл а за диспечиране на събития в приложението. На свой приложението възобновява обработката.	Няма	Статус

Таблица 2.16: Последователност от процеси при използване на VCI_Connect и VCI_Dispatch (доставяне на съобщение от VALUES).

2.7 Услуги за управление на възстановяванията

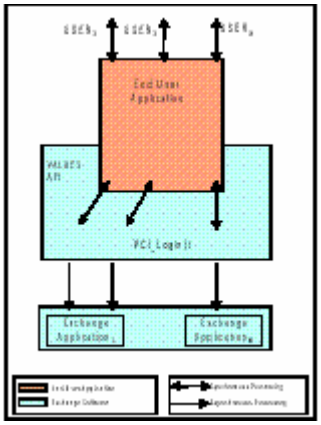
Някои Борсови приложения поддържат възстановяването на приложни заявки, приложни отговори и излъчвания. Подробни описания на тези услуги за управление на възстановяванията са дадени в томовете за съответните Борсови приложения.

Интерфейсът за повикване на VALUES API предлага обща услуга за възстановяване на прекъснати излъчвания. За тази цел е предвидена специална абонаментна тема, предназначена за предаване на пропуснати съобщения (Gap Notifications) за всеки поток, в който тази тема се използва. Как се извършва абониране за Gap Notifications обяснено в *раздел 3.9*.

Що се отнася до възстановимите излъчвания, за предпочитане е да се използват специфичните механизми за възстановяване на отделните Борсови приложения.

2.8 Възможност за поддържане на множество потребители

Една инстанция на VALUES API може да поддържа едновременен достъп на множество потребители до множество Борсови приложения.



Фигура 2.20: Достъп на множество потребители

Легенда:
.. Потребителско приложение
.. Борсов софтуер
... Синхронна обработка
... Асинхронна обработка

При влизане (по-конкретно с отговора на заявката за влизане), потребителското приложение получава от VALUES API идентификатор (login ID) на така създадения контекст на достъп (влизане в Борсовата услуга). Този идентификатор е уникален за всяко успешно повикване на VCI_Connect и за всяко отделно Борсово приложение. При последващи повиквания към входните точки VCI_Logout, VCI_Subscribe (за Xervices, които

поддържат Broadcast Extension, сравни с *точка 2.5.2*) и VCI_Submit, потребителското приложение е длъжно да посочва този login ID, за да идентифицира конкретния потребител.

Функцията за обратно повикване, която се регистрира с всяко повикване към VCI_Submit, VCI_Subscribe (само за Broadcast Extension, виж с *точка 2.5.2*) или VCI_Logout, връща login ID на потребителското приложение. На свой ред, с така получения login ID потребителското приложение може да разпределя отговорите по съответните потребители.

Всяко приложение, което поддържа множество потребители, отговаря за правилното и оторизирано използване на отделните login IDs, получавани от VALUES API.

2.9 Xervices, класове Xervices и възможност за поддържане на няколко Борси

Xervice е услуга, предоставяна от електронна борса на интерфейсна (клиентска) система, която е базирана на VALUES. Една борса може да предлага няколко Xervices, например Xetra Франкфурт предлага Xervice за сделки, Xervice за препредаване на (възстановими) излъчвания на данни и Xervice за котиране на непрекъснат аукцион. За да получи достъп до дадена Xervice, приложението трябва да влезне в нея чрез VCI_Login. Идентификаторът dbApplID (виж *точка 6.2.20*) определя към коя точно Xervice е насочено конкретното повикване.

Данните за идентифициране на Xervice са налични в оперативен режим, веднага щом има установена сесия с GATE. Предоставяните от GATE данни включват код за идентифициране на Борсата (Market Identification Code – MIC) и обяснителен текст. По този начин приложението може да избере борса или да предложи разбираема информация на потребителя. Това позволява на клиентските приложения да интегрират нови Борси, които не са били известни по време на развоя, без да се променя изпълнимия код. Това е валидно, само ако новата Борса предлага функционалност на VALUES, идентична на функционалността, предлагана от съществуваща Борса, за която приложението е било проектирано и разработено.

Клиентското приложение може да определи дали Xervices на различните електронни борси предлагат идентична функционалност за VALUES, като оцени данните за класа Xervice. Класът Xervice е идентификатор, който е единен за всички Xervices, предлагащи идентична функционалност на VALUES, при условие, че номерът на версията на приложението (Application Version Number) също е еднакъв. Например, ако дадено клиентско приложение е разработено така, че да работи с определена борса, като използва две Xervices, чиито Xervice класове са съответно XETRA_TXN_XCLASS и XETRA_CAQ_XCLASS с версия AVN_xyz, това приложение ще може да работи с всяка друга борса, която предлага същите класове Xervices и AVN. Идентификаторът на класа Xervice се дава в полето applClass на XerviceInfotT (виж *точка 6.2.2*).

GATE предоставя информация за класа и идентификационни данни за съответната Xervice чрез функцията за обратно повикване, регистрирана при използването на VCI_Connect. Повече подробности са дадени в *точка 3.3.2*.

2.10 Концепция за обратната съвместимост на VALUES API

В този раздел е обяснена в обща план концепцията за обратна съвместимост на VALUES API. Основните понятия във връзка с интерфейса за повикване са обяснени в този том, а тези за отделните борси – в томовете за съответните борси.

VALUES API поддържа обратна съвместимост, което означава, че поддържа както текущите версии, така и предишните версии на приложните заявки, структурите с излъчвани данни и интерфейса за повикване. Обратната съвместимост на интерфейса за повикване е независима от тази Борсовите приложения.

- Интерфейс за повикване (технически компоненти)
  VALUES API поддържа текущата версия на интерфейса за повикване (версия на интерфейса за повикване номер CVN_x), както и обратна съвместимост с предишната версия на интерфейса за повикване (версия номер CVN_{x-1}). Всяка входна точка от интерфейса за повикване трябва да има CVN, който дефинира коя версия на интерфейса следва да се използва. Всички входни точки на интерфейса за повикване, подадени по време на сесия във VALUES, трябва да съответстват на версията, посочена в заявката за свързване (connection request).

- Приложни заявки и заявки за абониране (функционални компоненти)
  VALUES API може да обслужва приложни заявки от текущата версия (AVN_x), като едновременно поддържа обратна съвместимост с предишната версия (AVN_{x-1}). Молим вижте томовете на отделните Борсови приложения, за да проверите дали желаното Борсово приложение предлага функционалност за обратна съвместимост. Всяка заявка за влизане или абониране трябва да съдържа номера на версията на Борсовото приложение (AVN), за да определи коя точно версия на Борсовото приложение следва да се използва. Всички приложни заявки, подадени след влизането в дадено Борсовото приложение, трябва да съответстват на версията, посочена в заявката за влизане (login request).

Номерът CVN се публикува с излизането на всяка нова версия на интерфейса за повикване на VALUES API. Номерът AVN ще се публикува с излизането на всяка нова версия на съответното Борсово приложение. Потребителското приложение трябва да посочва тези номера, за да определя коя версия възнамерява да използва, което означава, че CVN трябва да се изпраща при всяко повикване на входна точка, а AVN – при всяка заявка за влизане или абониране.

Обратната съвместимост не може да бъде гарантирана при всички обстоятелства поради възможността за промени, наложени от нормативните изисквания, развитието на технологиите и др. Това означава следното:

- Повечето актуализации на VALUES API няма да изискват повторно компилиране и привързване на потребителските приложения.

- Някои актуализации на VALUES API може да изискват повторно компилиране и привързване на потребителските приложения.

- Бъдещите версии на Борсовите приложения или GATE, както и промените на кода, евентуално могат да изискват повторно компилиране и привързване на потребителските приложения.

3 VALUES API. Интерфейс за повикване. Справочни данни

3.1 Обзор

В този раздел е описана структурата на всяка входна точка на интерфейса за повикване, включително име на точката, описание и синтаксис на командата за повикване. Даден е и списък на параметрите, от които са съставени входните точки. Параметрите могат да представляват структури, в който случай е даден списък на полетата.

За всяко поле в списъка е даден маркер, който посочва дали полето е задължително ("m" от mandatory), незадължително ("o" от optional), само за четене ("ro" от read-only) или резервирано ("r" от reserved). Тези маркери са дадени в колоната "Правило за използване". Освен това, в колоната "Попълва се от" има маркер, който посочва дали полето се попълва от VALUES ("V") или от потребителското приложение ("A"). Полетата, които се попълват от потребителските приложения, могат да бъдат променяни от VALUES. Символът "-" в едната или другата колона посочва, че информацията не е приложима за съответното поле.

Подробни описания на полетата са дадени в *раздел 6*. Описанията съдържат информация за това, къде се използва полето, както и някои характеристики на полето, например вид на данните и механизъм за предаване. Освен това описанията дефинират правилата, приложими към отделните полета с данни.

В този раздел са описани следните входни точки:

- VCI_Connect
- VCI_Disconnect
- VCI_Dispatch
- VCI_Login
- VCI_Logout
- VCI_Submit
- VCI_Subscribe
- VCI_Unsubscribe

По-нататък са спесифицирани функциите за обратно повикване. VALUES API предоставя функционален тип, който трябва да се използва за деклариране на функциите за обратно повикване, препратки към които се подават на VCI_Connect, VCI_Login, VCI_Logout, VCI_Submit и VCI_Subscribe.

Всички описани в документа интерфейсни параметри са указатели към структури от данни. Параметърът *callbackCntxData* (контекстуални данни за функцията за обратно

повикване), както и параметрите с приставка "req" (request, заявка) указват структури от данни, които потребителското приложение попълва и предава на входната точка от интерфейса за повикване.

Първият параметър на всяка входна точка е *reqControl*. Структурата *reqControl* включва полета от данни, които са общи за всички входни точки и се използват като информация за контрол на интерфейса за повикване.

Бележка: Настоятелно се препоръчва за достъп или копиране на полетата в reqControl винаги да се използва типа данни *ReqCntrlT*.

Параметърът *statusData* и параметрите с приставка "resp" (response, отговор) указват структури от данни, които се попълват от входната точка на интерфейса за повикване и се връщат на потребителското приложение. Изключения са функциите за обратно повикване. Подробно описание за попълването на структурите от данни на функциите за обратно повикване е дадено в *раздел 3.11*.

Повикващото приложение отговаря за предоставянето (и освобождаването) на достатъчно памет за съхранение на различните структури от данни. Подробности са дадени в *раздел 4*. Интерфейсът за повикване на VALUES API съхранява само указателите (pointers) към данните за заявките и контекста. Параметърът на ответното повикване се попълва с референция към функция. Сигнатурата на функцията трябва да съответства на спесификацията на интерфейса за обратно повикване (виж *раздел 3.11*).

Препоръчва се да се инициализират всички резервирани или неизползвани незадължителни полета с данни, които се подават към входните точки. Указания са дадени в *точка 6.1.2.*

Всяка входна точка връща към базираното на VALUES API приложение структура с данни за статуса, която съдържа информация за изпълнението на повикването. Полетата в структурата с данни за статуса са описани на следната таблица:

Източник на изключението	**Поле**	**Описание**
Борсово приложение	complSeverity	Сериозност (значимост на изключението)
	complCode	Уникален код на изключението в контекста на Борсовото приложение
	complText	Дешифровка, т.е. текстово представяне на изключението
GATE,	complSeverity	Сериозност (значимост на изключението)
VALUES API	complCode	Уникален код на изключението
	complText	Дешифровка, т.е. текстово представяне на изключението

Таблица 3.1: Структура с данни за статуса, предоставяни от интерфейса за повикване

Бележка: Статус за конкретното изпълнение от Борсово приложение се съобщава, само ако кодът за техническо изпълнение *(techComplCode)* показва неуспешно обработване на заявката.

В зависимост от значимостта на изключението, потребителското приложение трябва да изпълни различни видове обработки на изключения.

На следваща таблица са описани различните степени на значимост на изключенията със съответните кодове.

Поле	**Стойност**	**Описание**
complSeverity,	VCI_SUCCESS	Успешно изпълнение
techComplSeverity	VCI_WARNING	Възникнала е незначителна грешка. Може да се наложи приложението да извърши обработка на грешка в зависимост от това, какъв код за изпълнение е върнат.
	VCI_ERROR	Открита е грешка. Приложението задължително трябва да извърши обработка на грешка в зависимост от това, какъв код за изпълнение е върнат.
	VCI_FATAL	Възникнала е фатална грешка. Приложението трябва да изпълни процедурата за затваряне.

Таблица 3.2: Изключения, предоставяни от интерфейса за повикване. Класове на значимост.

За всяка входна точка е предоставен списък с приложимите кодове за изпълнение и текстовете на съответните съобщения. Пълен списък на кодовете за техническо изпълнение е даден в *раздел 5.* Кодовете за функционално изпълнение са дадени в томовете за отделните Борси.

3.2 Диаграми на състоянията

3.2.1. Обзор

Диаграмите на състоянията във VALUES API описват кои повиквания към VALUES API могат да бъдат активирани според текущото използване на VALUES API от приложението.

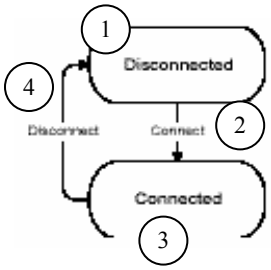
Преходите между състоянията могат да бъдат повиквания към VALUES API, активирани от потребителското приложение, или обратни повиквания към приложението, активирани от VALUES.

Нотацията 1..s означава множество възможни абонирания, 1..u означава множество потребители и 1..x означава използване на множество Xervices за съответните състояния.

В следващите точки са представени диаграмите на състоянията за всеки вид услуги на VALUES. Представени са диаграмите на състоянията при нормална работа и са описани изключенията.

3.2.2. Нормална работа

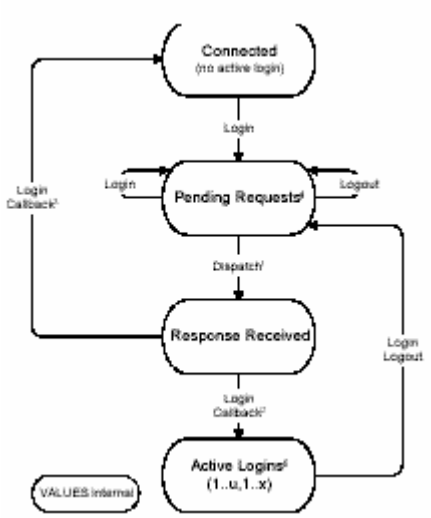
Услуги за управление на сесия



Фигура 3.1: Диаграма на състоянията при услугите за управление на сесия

Легенда:
1.....Състояние "Disconnected"
2.....Свързване
3.....Състояние "Connected"
4.....Изключване

Услуги за управление на сигурността



Фигура 3.2: Диаграма на състоянията при услугите за управление на сигурността

Легенда:

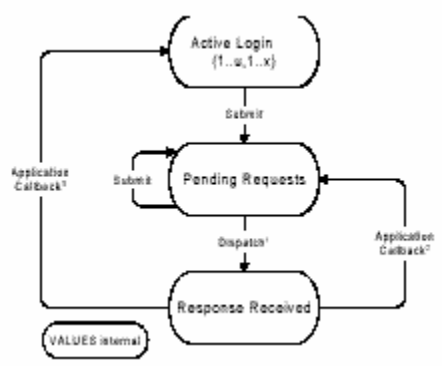
1 ..... Dispatch се повиква от приложението след известяване чрез VMQ.

2 ..... Функция за обратно повикване, активирана от VALUES при получаване на отговор на заявка за влизане или излизане.

3 ..... Функция за обратно повикване, активирана от VALUES при получаване на последния отговор на заявка за излизане (тоест, няма други активни влизания).

4 ..... Състоянията "Active Logins" (активни включвания) и "Pending Requests" (висящи заявки) могат да бъдат активни едновременно, в този случай са възможни преходи от двете състояния.

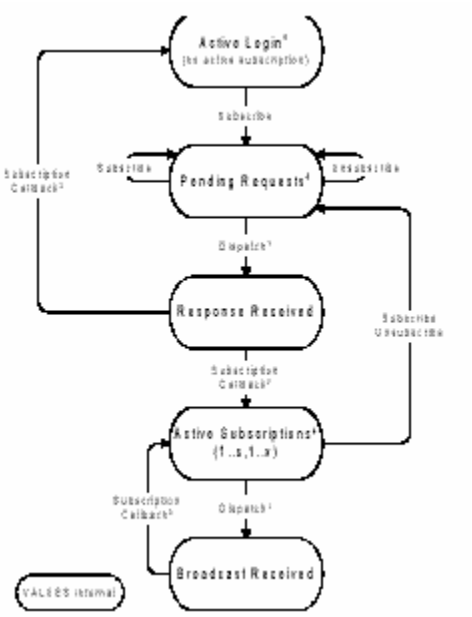
Услуги за управление на заявките



Фигура 3.3: Диаграма на състоянията при услугите за управление на заявките

Легенда:
1 ..... Dispatch се повиква от приложението след известяване чрез VMQ.
2 ..... Функция за обратно повикване, активирана от VALUES при получаване на отговор на приложна заявка (приложен отговор).
3 ..... Функция за обратно повикване, активирана от VALUES при получаване на последния приложен отговор (т.е. няма други висящи приложни заявки).

Услуги за управление на абонирания (приема се, че е реализирано разширението за излъчване на данни "Broadcast Extension")



Фигура 3.4: Диаграма на състоянията при услугите за управление на абонаменти

Легенда:

1 ..... Dispatch се повиква от приложението след известяване чрез VMQ.

2 ..... Функция за обратно повикване, активирана от VALUES при получаване на отговор на заявка за абониране или за прекратяване на абониране

3 ..... Функция за обратно повикване, активирана от VALUES при получаване на последния отговор на заявка за прекратяване на абониране (т.е. няма други активни абонаменти).

4 ..... Състоянията "Active Subscriptions" (активни абонаменти) и "Pending Requests" (висящи заявки) могат да бъдат активни едновременно, в този случай са възможни преходи от двете състояния.

5 ..... Функция за обратно повикване, активирана от VALUES при получаване на излъчване (емисия) от данни.

6 ..... Сравни с раздела за Broadcast Extension.

3.2.3. Обработка на изключения

При използването на VALUES API могат да възникнат редица изключения. При обработката на изключения VALUES API извършва смяна на състоянието, ако е приложимо, и известява потребителското приложение чрез връщане на статус или чрез активиране на една или няколко функции за обратни повиквания.

На следващите таблици са изведени основните случаи на изключения, включително състоянието във VALUES след обработката на съответното изключение. Посочен е и механизмът за известяване (включително techComplSeverity и techComplCode).

Изключение	**Резултативно състояние**
Потребителското приложение прекъсва връзката с VALUES, докато все още има висящи заявки, активни влизания или абонаменти	Disconnected

Механизъм за известяване:

Последователно се активират функциите за обратно повикване:

1. Login callback – за всеки активен контекст на влизане (login)
   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_XERVICE_NOT_AVAILABLE

2. Submit callback – за всяка висяща заявка (включително login, logout, subscribe и unsubscribe)
   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_PENDING_REQUEST_DELETED

3. Subscribe callback – за всеки активен абонамент
   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_SUBSCRIPTION_DELETED

4. Connection callback
   В зависимост от кода на изпълнението в отговора на заявката за излизане, това може да бъде:
   techComplSeverity: VCI_SUCCESS
   techComplCode: ELB_TECH_OK
   или
   techComplSeverity: VCI_ERROR
   techComplCode: ELB_TECH_REQ_UNSUCCESSFUL

Изключение	Резултативно състояние
Потребителското приложение излиза от дадена услуга, докато все още има висящи заявки	Connected
Механизъм за известяване: Последователно се активират функциите за обратно повикване: 1. Login callback – за контекста, създаден с влизането в тази услуга (т.е. за съответния login) techComplSeverity: VCI_SUCCESS techComplCode: ELB_TECH_LOGGED_OUT 2. Submit callback – за всяка висяща заявка в контекста на съответния login techComplSeverity: VCI_WARNING techComplCode: ELB_TECH_ PENDING_REQUEST_DELETED 3. Subscribe callback – за всяка висяща заявка за абониране в контекста на съответния login, ако съответната Xervice реализира Broadcast Extension (виж *точка 2.5.2*) 4. techComplSeverity: VCI_WARNING techComplCode: ELB_TECH_ PENDING_REQUEST_DELETED 5. Subscribe callback – за всеки активен абонамент в контекста на съответния login 6. techComplSeverity: VCI_WARNING techComplCode: ELB_TECH_SUBSCRIPTION _DELETED 7. Connection callback techComplSeverity: VCI_SUCCESS techComplCode: ELB_TECH_LOGGED_OUT	

Изключение	Резултативно състояние
Неправилно използване на входните точки (тоест преходът между състоянията не е показан в диаграмите на състоянията) или предаване на невалидни параметри до входните точки.	Без промяна
Механизъм за известяване: Входната точка връща статус, посочвайки изключението.	

Изключение	Резултативно състояние
Неуспешно валидиране на данните от приложна заявка.	Без промяна
Механизъм за известяване:	
Submit Callback се активира със статус, посочващ изключението.	

Изключение	Резултативно състояние
Неразполагаемост на Борсова услуга	Active Logins / Connected[1]
Механизъм за известяване:	

Последователно се активират функциите за обратно повикване:

1. Login callback – за всеки активен контекст на влизане (login) в засегнатата Борсова услуга
   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_XERVICE_NOT_AVAILABLE

2. Submit, Login, Subscribe (само заBroadcast Extension only) callback – за всяка висяща заявка към засегнатата Борсова услуга (включително submit, login, logout, subscribe и unsubscribe)
   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_ PENDING_REQUEST_DELETED

3. Subscription callback – за всеки активен абонамент на засегнатата Борса.

4. techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_ SUBSCRIPTION_DELETED

5. Connection callback
   techComplSeverity: VCI_SUCCESS
   techComplCode: ELB_TECH_XERVICE_NOT_AVAILABLE

Засегнатата Борсова услуга е посочена в dbAppIID на reqControl.

[1] Ако всички Борсови приложения са неразполагаеми, резултативното състояние е "Connected"

Изключение	**Резултативно състояние**
Неразполагаемост на GATE (Технически услуги)	Disconnected

Механизъм за известяване:

Последователно се активират функциите за обратно повикване:

1. Login callback – за всеки активен контекст на влизане (login)
   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_XERVICE_NOT_AVAILABLE

2. Submit callback – за всяка висяща заявка (включително login, logout, subscribe и unsubscribe)

   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_ PENDING_REQUEST_DELETED

3. Subscribe callback – за всеки активен абонамент
   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_SUBSCRIPTION_DELETED

4. Connection callback
   techComplSeverity: VCI_FATAL
   techComplCode: ELB_TECH_TECHSRCV_NOT_AVAILABLE

Изключение	Резултативно състояние
Непрозрачно прекъсване на MISS за Борсова услуга, която не поддържа препредаване на приложни заявки	Active Logins / Connected

Механизъм за известяване:

Последователно се активират функциите за обратно повикване:

1. Login callback – за всеки активен контекст на влизане (login) в съответната Борсова услуга
   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_NONTRANSPARENT_FAILOVER

2. Submit/Login/Subscribe callback – за всяка висяща заявка към конкретната Борсова услуга (включително submit, login, logout, subscribe и unsubscribe)
   techComplSeverity: VCI_WARNING
   techComplCode: ELB_TECH_ PENDING_REQUEST_DELETED

3. Connection callback
   techComplSeverity: VCI_SUCCESS
   techComplCode: ELB_TECH_NONTRANSPARENT_FAILOVER

Съответната Борсова услуга е посочена в dbApplID на reqControl.

3.3 VCI_Connect

3.3.1. Обзор

Описание

Тази входна точка на интерфейса за повикване служи за установяване на връзка с GATE. VCI_Connect инициира и осъществява връзка между приложението и VALUES с цел изпращане на заявки към Борсовите услуги и получаване на съответните отговори. VCI_Connect е входната точка, която инициира сесия във VALUES. След успешно изпълнение на VCI_Connect, приложението може да изпраща заявки за влизане в Борсовите услуги. При използване на Xervice, която не реализира Broadcast Extension (виж *точка 2.5.2*), има и възможност да се получи абонамент за излъчвания на потоци от данни, без преди това да е изпълнена процедура за влизане.

VCI_Connect изисква регистриране на функция за обратно повикване на приложението, която да получава известия и изключения (например неразполагаемост на Борсово приложение). Приложението може да спесифицира

индивидуални контекстуални данни за ответното повикване, които се буферират от VALUES и се връщат на приложението при активиране на функцията за обратно повикване. VALUES съхранява само адреса на контекстуалните данни, спесифицирани от потребителското приложение. В типичния случай VCI_Connect се активира при стартиране на приложението.

VCI_Connect е синхронна заявка. При успешно изпълнение VCI_Connect връща името на опашката за съобщения във VALUES (VMQname), която приложението трябва да наблюдава. За да не се прекъсва VCI_Connect, приложението трябва да забранява асинхронни събития (например сигнали от UNIX).

Забележка: VMQ трябва да бъде наблюдавана за събития веднага след успешното установяване на сесия. Това наблюдение трябва да се поддържа до прекратяване на сесията.

Описания на полетата на интерфейса за повикване е дадено в *раздел 6*.

Синтаксис

```
void VCI_Connect (
   ReqCntrlT          *reqControl,
   CnctReqDataT       *reqData,
   AppCallbackT       *callbackFunc,
   AppCntxtDataT      *callbackCntxtData,
   CnctRespDataT      *respData,
   StatusDataT        *statusDataGlobal
);
```

Наименование на параметъра	**Полета**	**Правило за използване**	**Попълва се от**
reqControl	VCIver (виж *раздел 3.3.2*)	m	A
	connectionID	r	-
	dbApplID	r	-
	loginID	r	-
	appDescr	r	-
	reqID	r	-
	resubmitFlag	r	-
	resubmitNo	r	-
reqData	userID	m	A
	password	m	A
callbackFunc	(Референция към функцията за обратни повиквания тип "изключение", виж *т. 3.3.3*)	m	A
callbackCntxtData	custBlockSize	m	A
	custData	o	A
respData	connectionID	ro	V
	VMQname	ro	V
	prodMode	ro	V
statusDataGlobal	complSeverity	ro	V
	complCode	ro	V
	complText	ro	V
	techComplSeverity	ro	V
	techComplCode	ro	V
	techComplText	ro	V

Наименование на полето за статус	**Стойност**	**Текст**
complSeverity, techComplSeverity	VCI_SUCCESS	
	VCI_WARNING	
	VCI_ERROR	
	VCI_FATAL	
complCode, complText	ELB_TECH_INVALID_PASSWORD	НЕВАЛИДНА ПАРОЛА
	ELB_TECH_INVALID_USER	ПОТРЕБИТЕЛЯТ НЯМА ПРАВО ДА ИЗПОЛЗВА БОРСОВАТА УСЛУГА ИЛИ ПОТРЕБИТЕЛЯТ НЕ Е РЕГИСТРИРАН
	ELB_TECH_INVALID_USER_OR_PASSWORD	НЕВАЛИДЕН ПОТРЕБИТЕЛ ИЛИ ПАРОЛА(1)
	ELB_TECH_BADLY_FORMED_PARAMETER	ПАРАМЕТЪРЪТ Е НЕВАЛИДЕН ИЛИ СЪДЪРЖА НЕВАЛИДНИ СИМВОЛИ[a]
techComplCode, techComplText	ELB_TECH_OK	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_INVALID_PARAMETER	ПРЕДАДЕН Е НЕВАЛИДЕН ПАРАМЕТЪР
	ELB_TECH_INTERNAL_ERROR	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_ALREADY_CONNECTED	ПРИЛОЖЕНИЕТО ВЕЧЕ Е СВЪРЗАНО
	ELB_TECH_CONNECTION_LIMIT_REACHED	ДОСТИГНАТ Е МАКСИМАЛНИЯТ БРОЙ ВРЪЗКИ
	ELB_TECH_REQ_UNSUCCESSFUL	НЕУСПЕШНО ИЗПЪЛНЕНИЕ НА ЗАЯВКАТА
	ELB_TECH_TECHSRV_NOT_AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА ТЕХНИЧЕСКИТЕ УСЛУГИ
	ELB_TECH_XERVICE_NOT_AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА БОРСОВАТА УСЛУГА
	ELB_TECH_NOT_REENTRANT	ПОВИКВАНИЯТА КЪМ VALUES НЕ СА ПОВТОРЯЕМИ

(1) Важи само за платформи Windows.

3.3.2. VALUES API. Версия на интерфейса за повикване

В концептуално отношение VALUES API поддържа съвместимост на интерфейса за повикване. В настоящия документ е описана версия 1.2 на този интерфейс. Стойността Vresion Clver на интерфейса за повикване на VALUES API може да бъде зададена като CVN_xxx, където "xxx" зависи от версията, която потребителското приложение възнамерява да използва. Специфичната константа CVN_xxx е дефинирана в заглавния (header) файл VALUES.h, който се предоставя с пакета от заглавни файлове за VALUES API.

Константната стойност на версията на интерфейса за повикване, която трябва да се използва за GATE Release 3.5, е CVN_012. Това е същата версия, която е в употреба след GATE Release 3.1.

Бележка: По време на една сесия във VALUES трябва да се използва само един и същ CVN за всички входни точки. Това се проверява от VALUES.

3.3.3. Функцията connect callback

Функцията connect callback се използва за известяване (нотифициране) на приложението относно настъпили изключения (напр. неразполагаемост на GATE). Освен това, тя се използва за уведомяване на потребителското приложение относно промените в разполагаемостта на Борсовите приложение (тоест дали дадена Xervice работи или не работи). След успешно изпълнение на VCI_Connect, функцията connect callback се активира (извиква) от VALUES за всяка Борсова услуга, която е в състояние на разполагаемост.

Данни за функционалните прототипи, параметрите и кодовете за изпълнение на функцията connect callback са дадени в *раздел 3.11*.

Известията за изключения и промени на разполагаемостта се връщат с данните за статуса на функцията connect callback. При настъпване на събитие, свързано с дадена Xervice (виж *раздел 2.9* за концептуално описание на Xervice), следната информация се предава на функцията за обратно повикване (като указател към структура от вида XerviceInfoT):

Наименование на параметъра	**Полета**	**Правило за използване**	**Попълва се от**
appRespData	applClass	-	V
	applVersion	-	V
	applPrevVersion	-	V
	exchApplId	-	V
	exchDscrName	-	V

Това се отнася за обратни повиквания, активирани със следните кодове за изпълнение (techComplCod):

- ELB_TECH_XERVICE_AVAILABLE
- ELB_TECH_XERVICE_NOT_AVAILABLE
- ELB_TECH_LOGGED_OUT
- ELB_TECH_NONTRANSPARENT_FAILOVER

Трябва да се има предвид, че тези случаи параметърът dbApplID, който посочва коя точно Xervice е задействала събитието, винаги е наличен от структурата за контрол на заявката (reqControl->dbApplID).

3.4 VCl_Disconnect

Описание

Тази входна точка на интерфейса за повикване се използва от потребителското приложение за прекратяване на сесията във VALUES. В типичния случай VCl_Diconnect се повиква при прекратяване на сесия чрез затваряне на потребителското приложение.

VCl_Diconnect е синхронна заявка. За да не се прекъсва VCl_Disconnect, приложението трябва да забранява асинхронни събития (например сигнали от UNIX).

Синтаксис

```
void VCl_Disconnect(
        ReqCntrlT           *reqControl,
        DiscnctReqDataT     *reqData,
        StatusDataT         *statusDataGlobal
        );
```

Наименование на параметъра	Полета	Правило за използване	Попълва се от
reqControl	VCIver	m	A
	connectionID	m	A
	dbApplID	r	-
	loginID	r	-
	appDescr	r	-
	reqID	r	-
	resubmitFlag	r	-
	resubmitNo	r	-

Наименование на параметъра	Полета	Правило за използване	Попълва се от
reqData	n/a	r	-
statusDataGlobal	complSeverity	r	-
	complCode	r	-
	complText	r	-
	techComplSeverity	ro	V
	techComplCode	ro	V
	techComplText	ro	V

Наименование на полето за статус	Стойност	Текст
complSeverity	n/a	
techComplSeverity	VCI_SUCCESS	
	VCI_WARNING	
	VCI_ERROR	
	VCI_FATAL	
complCode, complText	n/a	
techComplCode, techComplText	ELB_TECH_OK	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_INVALID_PARAMETER	ПРЕДАДЕН Е НЕВАЛИДЕН ПАРАМЕТЪР
	ELB_TECH_INTERNAL_ERROR	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_NOT_ CONNECTED	ПРИЛОЖЕНИЕТО НЕ Е СВЪРЗАНО
	ELB_TECH_NOT_REENTRANT	ПОВИКВАНИЯТА КЪМ VALUES НЕ СА ПОВТОРЯЕМИ

3.5 VCI_Dispatch

Описание

Тази входна точка на интерфейса за повикване се използва от потребителското приложение за обработка на отговори от Борсовите услуги, излъчвания, известия и изключения. Потребителското приложение е длъжно да проверява VMQ за настъпили събития. При настъпване на събитие, приложението повиква VCI_Dispatch. На свой ред VCI_Dispatch прочита данните от опашката за съобщения във VALUES и ги предава на съответната функция за обратно повикване. VCI_Dispatch е синхронна заявка, която се обръща към повикващото потребителско приложение след като завърши активирането на функцията (функциите) за обратно повикване.

VCI_Dispatch разбира коя функция на обратно повикване да активира по вътрешна таблица във VALUES, която съхранява всички свързвания, влизания, заявки и абонирания на приложението, както и съответните функции за обратно повикване. Повече информация относно начина за интегриране на събитията във VALUES е дадена в *раздел 2.6.*

Синтаксис

```
void VCI_Dispatch(
    ReqCntrlT       *reqControl,
    StatusDataT     *statusDataGlobal
    );
```

Наименование на параметъра	Полета	Правило за използване	Попълва се от
reqControl	VCIver	m	A
	connectionID	m	A
	dbApplID	r	-
	loginID	r	-
	appDescr	r	-
	reqID	r	-
	resubmitFlag	r	-
	resubmitNo	r	-
statusDataGlobal	complSeverity	r	-
	complCode	r	-
	complText	r	-

Наименование на параметъра	Полета	Правило за използване	Попълва се от
	techComplSeverity	ro	V
	techComplCode	ro	V
	techComplText	ro	V

Наименование на полето за статус	Стойност	Текст
complSeverity	n/a	
techComplSeverity	VCI_SUCCESS	
	VCI_WARNING	
	VCI_ERROR	
	VCI_FATAL	
complCode, complText	n/a	
techComplCode, techComplText	ELB_TECH_OK	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_INVALID_PARAMETER	ПРЕДАДЕН Е НЕВАЛИДЕН ПАРАМЕТЪР
	ELB_TECH_INTERNAL_ERROR	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_NOT_ CONNECTED	ПРИЛОЖЕНИЕТО НЕ Е СВЪРЗАНО
	ELB_TECH_MSG_DISCARDED	ОТСТРАНЕНО Е НЕРАЗПОЗНАВАЕМО СЪОБЩЕНИЕ
	ELB_TECH_NOT_REENTRANT	ПОВИКВАНИЯТА КЪМ VALUES НЕ СА ПОВТОРЯЕМИ

3.6 VCI_Login

Описание

Тази входна точка на интерфейса за повикване се използва от потребителското приложение за влизане в Борсовите услуги. Преди да изпраща заявки към Борсовите приложения, потребителят трябва да получи оторизация от съответната Борсова услуга. Това става чрез повикване на VCI_Login за всяка желана Борсова услуга. Тази заявка е асинхронна. Xervices, които реализират Broadcast Extension (виж *раздел 2.5.2*), изискват също така оторизация при абонирането за излъчвания на потоци от данни.

VCI_Login изисква да се регистрира функция за обратно повикване на приложението. Тази функция (login callback) се използва при получаване на отговори на заявките за влизане и излизане, както и за известяване на потребителското приложение относно възникнали случаи на изключения (например неизправност на Борсова услуга).

Отговорът на заявката за влизане се получава асинхронно през регистрираната функция за обратно повикване. Статусът *(techComplCode)* на тази функция изрично показва получаването на отговори на заявки за влизане и излизане.

Потребителското приложение може да повиква VCI_Dispatch многократно, за да осигури влизане на различни потребители. С отговора на заявката за влизане се връща уникален идентификатор *(loginID)* на всеки успешно създаден конеткст на влизане. Така върнатият *loginID* задължително се посочва при всяко следващо повикване на VCI_Submit или VCI_Logout, за да осигури оторизация на конкретния потребител. При Xervices, които реализират Broadcast Extension (виж раздел *2.5.2*), това се отнася и за VCI_Subscribe.

Синтаксис

```
void VCI_Login(
    ReqCntrlT       *reqControl,
    LoginReqDataT   *reqData,
    AppCallbackT    *callbackFunc,
    AppCntxtDataT   *callbackCntxtData,
    LoginRespDataT  *respData,
    StatusDataT     *statusDataGlobal
    );
```

Наименование на параметъра	**Полета**	**Правило за използване**	**Попълва се от**
reqControl	VCIver	m	A
	connectionID	m	A
	dbApplID	m	A
	loginID	r	-
	appDescr	r	-
	reqID	r	-
	resubmitFlag	r	-
	resubmitNo	r	-
reqData[1]	userID	m	A
	applVersion	m	A
	authorizationData	m	A
	authorizationDataLength	m	A
callbackFunc	(референция към съответната функция за обратно повикване на приложението)	m	A
callbackCntxtData	custBlockSize	m	A
	custData	o	A
respData	n/a	r	-
statusDataGlobal	complSeverity	r	-
	complCode	r	-
	complText	r	-
	techComplSeverity	ro	V
	techComplCode	ro	V
	techComplText	ro	V

[1] За подробности виж томовете за съответните Борсови приложения.

Наименование на полето за статус	**Стойност**	**Текст**
complSeverity	n/a	
techComplSeverity	VCI_SUCCESS	
	VCI_WARNING	
	VCI_ERROR	
	VCI_FATAL	
complCode,	n/a	
complText		
techComplCode,	ELB_TECH_OK	УСПЕШНО ИЗПЪЛНЕНИЕ
techComplText	ELB_TECH_INVALID_PARAMETER	ПРЕДАДЕН Е НЕВАЛИДЕН ПАРАМЕТЪР
	ELB_TECH_INTERNAL_ERROR	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_NOT_ CONNECTED	ПРИЛОЖЕНИЕТО НЕ Е СВЪРЗАНО
	ELB_TECH_LOGIN_NUM_ EXCEEDED	ПРЕВИШЕН Е РАЗРЕШЕНИЯТ МАКСИМАЛЕН БРОЙ ВЛИЗАНИЯ
	ELB_TECH_XERVICE_NOT_ AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА БОРСОВАТА УСЛУГА
	ELB_TECH_TOO_MANY_ PENDING_REQUESTS	В ОПАШКАТА ИМА ПРЕКАЛЕНО МНОГО ВИСЯЩИ ЗАЯВКИ
	ELB_TECH_NOT_REENTRANT	ПОВИКВАНИЯТА КЪМ VALUES НЕ СА ПОВТОРЯЕМИ

3.6.1. Функцията login callback

Функцията login callback се използва за получаване на отговори на заявки за влизане и излизане, както и за известяване на потребителското приложение относно изключенията (например неразполагаемост на Борсова услуга). Данни относно функционалните прототипи, параметрите и кодовете за изпълнение на функциите за обратно повикване са дадени в *раздел 3.11*. За отговорите на заявките за влизане, функцията за обратно повикване връща следните данни *(LoginRespdataT)*:

Наименование на параметъра	Полета	Правило за използване	Попълва се от
applrespData	functResult	r	-
	loginID	ro	V

Статусът на отговорите на заявките за влизане и излизане, както и известията за изключения се връщат с данните за статуса на функцията login callback.

3.7 VCI_Logout

Описание

Тази входна точка на интерфейса за повикване се използва от потребителското приложение за излизане от Борсовите услуги.

VCI_Logout е асинхронна заявка.

VCI_Logout извършва излизане само за конкретен loginID, т.е. за контекста на влизането, реализирано от конкретен потребител.

VALUES препраща заявката за излизане на посочената Борсова услуга. Отговорът на заявката за излизане се предава асинхронно на функцията за обратно повикване на приложението, която е била регистрирана със съответната заявка за влизане. Получаването на отговорите на заявките за излизане се посочва изрично чрез статуса *(techComplCode)* на функцията за обратно повикване. Повече данни относно функцията login callback са дадени в *раздел 3.11* и *точка 3.6.1.*

Синтаксис

```
void VCI_Logout(
        ReqCntrlT           *reqControl,
        LogoutReqDataT      *reqData,
        LogoutRespDataT     *respData,
        StatusDataT         *statusDataGlobal
        );
```

Наименование на параметъра	Полета	Правило за използване	Попълва се от
reqControl	VCIver	m	A
	connectionID	m	A
	dbApplID	m	A
	loginID	m	A
	appDescr	r	-

Наименование на параметъра	Полета	Правило за използване	Попълва се от
	reqID	r	-
	resubmitFlag	r	-
	resubmitNo	r	-
reqData	n/a	r	-
respData	n/a	r	-
statusDataGlobal	complSeverity	r	-
	complCode	r	-
	complText	r	-
	techComplSeverity	ro	V
	techComplCode	ro	V
	techComplText	ro	V

Наименование на полето за статус	Стойност	Текст
complSeverity	n/a	
techComplSeverity	VCI_SUCCESS	
	VCI_WARNING	
	VCI_ERROR	
	VCI_FATAL	
complCode, complText	n/a	
techComplCode, techComplText	ELB_TECH_OK	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_INVALID_PARAMETER	ПРЕДАДЕН Е НЕВАЛИДЕН ПАРАМЕТЪР
	ELB_TECH_INTERNAL_ERROR	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_NOT_ CONNECTED	ПРИЛОЖЕНИЕТО НЕ Е СВЪРЗАНО

Наименование на полето за статус	Стойност	Текст
	ELB_TECH_TOO_MANY_ PENDING_REQUESTS	В ОПАШКАТА ИМА ПРЕКАЛЕНО МНОГО ВИСЯЩИ ЗАЯВКИ
	ELB_TECH_NOT_LOGGED_IN	ПОТРЕБИТЕЛЯТ НЕ Е ПОЛУЧИЛ ДОСТЪП
	ELB_TECH_NOT_REENTRANT	ПОВИКВАНИЯТА КЪМ VALUES НЕ СА ПОВТОРЯЕМИ

3.8 VCI_Submit

Описание

Тази входна точка на интерфейса за повикване се използва от потребителското приложение за изпращане на заявки за едни или други обработки към Борсовите услуги. Потребителското приложение спесифицира идентификатор на заявката (request ID), данни за заявката и функция за обратно повикване. Приложението може да спесифицира индивидуални контекстуални данни за ответното повикване като указател, който се връща на приложението при активиране на функцията за обратно повикване. VALUES съхранява само указатели към заявката и индивидуалните контекстуални данни, но не съхранява самите данни. VALUES препраща заявката на съответната Борсова услуга. Отговорът от Борсовата услуга се маршрутизира обратно до заявителя чрез спесифицираната от приложението функция за обратно повикване.

При повикване на VCI_Submit задължително се посочва *loginID* с цел идентифициране на потребителя. Този *loginID* трябва да е получен преди това чрез VCI_Login.

VCI_Submit е асинхронна входна точка.

Синтаксис

```
void VCI_Submit(
ReqCntrlT               *reqControl,
SubmitReqDataT          *reqData,
        AppCallbackT    *callbackFunc,
        AppCntxtDataT   *callbackCntxtData,
StatusDataT             *statusDataGlobal
);
```

Наименование на параметъра	Полета	Правило за използване	Попълва се от
reqControl	VCIver	m	A
	connectionID	m	A

Наименование на параметъра	Полета	Правило за използване	Попълва се от
	dbApplID	m	A
	loginID	m	A
	appDescr	r	-
	reqID	m	A
	resubmitFlag	r	-
	resubmitNo	o(1)	A
reqData	appReqBlockSize	m	A
	appReq	m	A
callbackFunc	(референция към функцията за обратно повикване)	m	A
callbackCntxtData	custBlockSize	m	A
	custData	o	A
statusDataGlobal	complSeverity	r	-
	complCode	r	-
	complText	r	-
	techComplSeverity	ro	V
	techComplCode	ro	V
	techComplText	ro	V

(1) Някои Борсови приложения поддържат повторно подаване на заявки към тях. Подробности са дадени в томовете за съответните приложения.

Наименование на полето за статус	Стойност	Текст
complSeverity	n/a	
techComplSeverity	VCI_SUCCESS	
	VCI_WARNING	
	VCI_ERROR	
	VCI_FATAL	

Наименование на полето за статус	Стойност	Текст
complCode, complText	n/a	
techComplCode, techComplText	ELB_TECH_OK	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_INVALID_PARAMETER	ПРЕДАДЕН Е НЕВАЛИДЕН ПАРАМЕТЪР
	ELB_TECH_INTERNAL_ERROR	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_NOT_ CONNECTED	ПРИЛОЖЕНИЕТО НЕ Е СВЪРЗАНО
	ELB_TECH_TOO_MANY_ PENDING_REQUESTS	В ОПАШКАТА ИМА ПРЕКАЛЕНО МНОГО ВИСЯЩИ ЗАЯВКИ
	ELB_TECH_NOT_LOGGED_IN	ПОТРЕБИТЕЛЯТ НЕ Е ПОЛУЧИЛ ДОСТЪП
	ELB_TECH_NOT_REENTRANT	ПОВИКВАНИЯТА КЪМ VALUES НЕ СА ПОВТОРЯЕМИ

3.8.1. Функцията submit callback

Тази функция за обратно повикване се използва за предаване до приложението на отговори и изключения (например за неразполагаемост на GATE). Данни относно функционалните прототипи, параметрите и кодовете за изпълнение на функциите за обратно повикване са дадени в *раздел 3.11*.

Наименование на параметъра	Полета	Правило за използване	Попълва се от
applRespData	Виж томовете за съответните Борсови приложения	-	-

Статусът на обработката на заявките, както и известията за изключения се връщат с данните за статуса на функцията submit callback.

3.9 VCI_Subscribe

Описание

Тази входна точка на интерфейса за повикване се използва от потребителското приложение при абониране за потоци от данни. Преди да се абонира за даден поток от данни, приложението трябва успешно да е установило сесия (чрез VCI_Connect). Тези Xervices, които реализират Broadcast Extension, изискват и оторизация чрез login. При повикване на VCI_Subscribe приложението трябва да посочи желания поток от данни и да регистрира функция за обратно повикване. Приложението може да спесифицира индивидуални контекстуални данни за ответното повикване, които се връщат на приложението при активиране на функцията за обратно повикване. VALUES съхранява само указатели към заявката и индивидуалните контекстуални данни.

VCI_Subscribe е асинхронна заявка, която може да предизвика няколко вида отговори: отговор на заявката за абониране и отговор с абонаментни данни. Отговорът на заявката за абониране се получава асинхронно чрез регистрираната функция за обратно повикване на приложението. Получаването на отговори на заявките за абониране се показва изрично чрез статуса *(techComplCode)* на функцията за обратно повикване. Абонаментът е активен само след получаване на отговор "успешно абониране".

Данните, които се получават по време на активния абонамент, се предават на потребителското приложение чрез същата регистрирана функция за обратно повикване. Потребителското приложение разпознава дали получените данни са по абонамент или представляват отговори на заявки за абониране/прекратяване на абонирането чрез данните за статуса (techComplCode) на функцията за обратно повикване. Виж *раздел 3.11* за конкретния techComplCode.

Предвидена е специална абонаментна тема за предаване на пропуснати съобщения (Gap Notifications) за всеки поток, в който тази тема се използва. За тази цел, на reqData.subject, reqData.subjectLength и reqData.applVersion трябва да се зададат специалните стойности SUBJECT_GAPINFO, strlen(SUBJECT_GAPINFO) и SUBJECT_GAPINFO_VERSION. Молим да се има предвид, че абонирането с метасимвол (wildcard) не е достатъчно за получаване на пропуснати съобщения. При получаване на пропуснати съобщения в регистрираната от VCI_Subscribe функция за обратно повикване, темата се поставя на SUBJECT_GAPINFO. Ако повече няма нужда от получаване на пропуснати съобщения, абонирането трябва да се прекрати по нормалния ред.

Ако самите излъчвания са възстановими, използването на процедурата, предлагана от конкретното Борсово приложение (Broadcast Retransmission) е за предпочитане пред механизма с използване на SUBJECT_GAPINFO.

Синтаксис

```
void VCI_Subscribe(
        ReqCntrlT       *reqControl,
        SubsReqDataT    *reqData,
        AppCallbackT    *callbackFunc,
        AppCntxtDataT   *callbackCntxtData,
        SubsRespDataT   *respData,
        StatusDataT     *statusDataGlobal );
```

Наименование на параметъра	**Полета**	**Правило за използване**	**Попълва се от**
reqControl	VCIver	m	A
	connectionID	m	A
	dbApplID	m	A
	loginID	m/r	A, само за Xervices с Broadcast Extension, виж *точка 2.5.2*.
	appDescr	r	-
	reqID	r	-
	resubmitFlag	r	-
	resubmitNo	r	-
reqData	subsSubject	r	-
	applVersion(1)	m	A
	streamType	m	A
	subjectLength	m	A
	subject(2)	m	A
	userID	r	-
	authorizationDataLength	r	-
	authorizationData	r	-
callbackFunc	(референция към функцията за обратно повикване)	m	A
callbackCntxtData	custBlockSize	m	A
	custData	o	A

Наименование на параметъра	Полета	Правило за използване	Попълва се от
respData	n/a	r	-
statusDataGlobal	complSeverity	r	-
	complCode	r	-
	complText	r	-
	techComplSeverity	ro	V
	techComplCode	ro	V
	techComplText	ro	V

(1) Абонирането за пропуснати съобщения от потока става, като се зададат следните стойности: applVersion = SUBJECT_GAPINFO_VERSION, subject = SUBJECT_GAPINFO и subjectLength = strlen(SUBJECT_GAPINFO).

(2) Структурите на темите са дефинирани в томовете на съответните Борсови приложения за всяка заявка за абониране.

Наименование на полето за статус	Стойност	Текст
complSeverity	n/a	
techComplSeverity	VCI_SUCCESS	
	VCI_WARNING	
	VCI_ERROR	
	VCI_FATAL	
complCode, complText	n/a	
techComplCode, techComplText	ELB_TECH_OK	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_INVALID_PARAMETER	ПРЕДАДЕН Е НЕВАЛИДЕН ПАРАМЕТЪР
	ELB_TECH_INTERNAL_ERROR	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_NOT_ CONNECTED	ПРИЛОЖЕНИЕТО НЕ Е СВЪРЗАНО
	ELB_TECH_TOO_MANY_	В ОПАШКАТА ИМА ПРЕКАЛЕНО

Наименование на полето за статус	Стойност	Текст
	PENDING_REQUESTS	МНОГО ВИСЯЩИ ЗАЯВКИ
	ELB_TECH_NOT_LOGGED_IN	ПОТРЕБИТЕЛЯТ НЕ Е ПОЛУЧИЛ ДОСТЪП (само за Xervices с Broadcast Extension, виж *точка 2.5.2*.)
	ELB_TECH_NOT_REENTRANT	ПОВИКВАНИЯТА КЪМ VALUES НЕ СА ПОВТОРЯЕМИ

3.9.1. Функцията subscribe callback

Функцията subscribe callback се използва за получаване на отговори на заявките за абониране. Данни относно функционалните прототипи, параметрите и кодовете за изпълнение на функциите за обратно повикване на приложенията са дадени в *раздел 3.11*.

По заявките за абониране, функцията за обратно повикване на приложението връща следните данни *(SubsRespDataT)*:

Наименование на параметъра	Полета	Правило за използване	Попълва се от
applRespData	subsID	ro	V

Статусът на отговора на заявката за абониране, както и известията за изключения се връщат с данните за статуса на функцията subscribe callback.

В зависимост от активните абонаменти, излъчванията могат да са много чести. За да може приложението да получава тези излъчвания достатъчно бързо и за да се предпази VMQ от препълване, основната цел на дизайна трябва да бъде реализирането на стегнати обратни повиквания по време на абонамента. Ако се очаква обработката на излъчваните данни да отнема значително време, това трябва да става по начин, който не забавя получаването на следващите излъчвания. Ако VMQ се препълни, GATE "изхвърля" приложението от VALUES и сесията се прекратява.

3.10 VCI_Unsubscribe

Описание

Тази входна точка на интерфейса за повикване се използва от потребителското приложение при прекратяване на абонамента за посочения поток от данни. Прекратяването на абонамента води до изключване на потока от данни и изтриване на референцията към функцията за обратно повикване, която дотогава се поддържа във VALUES.

VCI_Unsubscribe е асинхронна заявка.

Отговорът на заявката за прекратяване на абонамента се получава чрез функцията за обратно повикване на приложението, която е била регистрирана със съответната заявка за абониране. Получаването на отговори на заявките за прекратяване на абонирането се показва изрично чрез статуса *(techComplCode)* на функцията за обратно повикване.

За повече подробности относно функцията subscribe callback виж *точка 3.9.1*.

Синтаксис

```
void VCI_Unsubscribe(
        ReqCntrlT           *reqControl,
        UnsubsReqDataT      *reqData,
        UnsubsRespDataT     *respData,
        StatusDataT         *statusDataGlobal
        );
```

Наименование на параметъра	**Полета**	**Правило за използване**	**Попълва се от**
reqControl	VCIver	m	A
	connectionID	m	A
	dbApplID	r	-
	loginID	r	-
	appDescr	r	-
	reqID	r	-
	resubmitFlag	r	-
	resubmitNo	r	-
reqData	subsID	m	A
respData	n/a	r	-
statusDataGlobal	complSeverity	r	-
	complCode	r	-
	complText	r	-
	techComplSeverity	ro	V
	techComplCode	ro	V
	techComplText	ro	V

Наименование на полето за статус	**Стойност**	**Текст**
complSeverity	n/a	
techComplSeverity	VCI_SUCCESS	
	VCI_WARNING	
	VCI_ERROR	
	VCI_FATAL	
complCode, complText	n/a	
techComplCode, techComplText	ELB_TECH_OK	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_INVALID_PARAMETER	ПРЕДАДЕН Е НЕВАЛИДЕН ПАРАМЕТЪР
	ELB_TECH_INTERNAL_ERROR	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_NOT_ CONNECTED	ПРИЛОЖЕНИЕТО НЕ Е СВЪРЗАНО
	ELB_TECH_TOO_MANY_ PENDING_REQUESTS	В ОПАШКАТА ИМА ПРЕКАЛЕНО МНОГО ВИСЯЩИ ЗАЯВКИ
	ELB_TECH_NOT_SUBSCRIBED	ПОТРЕБИТЕЛЯТ НЕ Е АБОНИРАН
	ELB_TECH_NOT_REENTRANT	ПОВИКВАНИЯТА КЪМ VALUES НЕ СА ПОВТОРЯЕМИ

3.10 Типове функции за обратно повикване на приложението

Наименование на типа: AppCallbackT

Описание:

Функциите за обратно повикване от този тип (application callback) са реализирани в потребителското приложение и се извикват (активират) от VALUES за извършване на асинхронни обработки в отговор на приложни заявки (включително login, logout, subscribe и unsubscribe), абонирания и изключения. Всички функции за обратни повиквания имат общ интерфейс и могат да бъдат регистрирани през входните точки VCI_Connect, VCI_Login, VCI_Logout, VCI_Submit и VCI_Subscribe.

Приложението може да спесифицира индивидуални контекстуални данни за ответното повикване като указател, който се връща на приложението при активиране на функцията за обратно повикване. VALUES съхранява само указатели към заявката и индивидуалните контекстуални данни, но не съхранява самите данни. И двете полета се освобождават (де-алоцират) от потребителското приложение. Функциите от тип application callback се повикват синхронно чрез VCI_Dispatch.

Всички структури с данни се заделят (алоцират) и попълват от приложението. Структурата statusData предава на потребителското приложение статус на обработката на заявката, статус на абонамента или данни за изключение.

Структурата с данни, указвана с параметъра appData, съдържа данни за приложните отговори и заявки. Указателят към първоначалните данни на приложната заявка се предава обратно на приложението.

Забележка: Функцията за обратно повикване не може да излъчва повиквания към входните точки на VALUES API. Ако повикване към входна точка на VCI бъде подадено от функция за обратно повикване, активирана с VCI_Dispatch, това повикване ще бъде отказано с ELB_TECH_NOT_REENTRANT.

Синтаксис:

```
typedef void (AppCallbackT) (
        ReqCntrlT           *reqControl,
        CallBkAppDataT      *appData,
        AppCntxtDataT       *callbackCntxtData,
        StatusDataT         *statusDataGlobal
        );
```

Наименование на параметъра	Полета	Правило за използване	Попълва се от
reqControl	VCIver	r	-
	connectionID	r	-
	dbApplID(1)	ro	V
	loginID(1)	ro	V
	appDescr	r	-
	reqID	ro	V
	resubmitFlag(2)	ro	V
	resubmitNo(2)	ro	V
appData	appReqBlockSize	ro	V
	appReq	ro	V
	subsID	ro	V (използва се само за subscription callback)
	subject	r	-
	appReqBlockSize	ro	V (използва се само за request callback)
	appReqData	ro	V
	applVersion	ro	V (с изключение на ответното повикване за VCI_Connect)
	streamType	ro	V (използва се само за subscription callback)
	brcSubject(3)	ro	V (използва се само за subscription callback)
callbackCntxtData	custBlockSize	ro	V
	custData	ro	V
statusDataGlobal	complSeverity	ro	V
	complCode	ro	V
	complText	ro	V
	techComplSeverity	ro	V
	techComplCode	ro	V
	techComplText	ro	V

(1) Винаги зададено, освен при connection callback с techComplCode: ELB_TECH_OK, ELB_TECH_TECHSRVC_NOT_AVAILABLE, ELB_TECH_REQ_UNSUCCESSFUL

(2) Задава се само при login callback (с изключение на отговор на заявка за влизане), connection callback с techComplCode: ELB_TECH_LOGGED_OUT и submission callback.

(3) Структурата на brcSubject е дефинирана в томовете на съответните Борсови приложения за всяка заявка за абониране.

Наименование на полето за статус	**Стойност**
complSeverity	
techComplSeverity	VCI_SUCCESS VCI_WARNING VCI_ERROR VCI_FATAL
complCode, complText	Тези полета връщат статус на изпълнението от съответното Борсово приложение. Подробни списъци на кодовете за изпълнение, генерирани от отделните Борсови приложения, са дадени в съответните томове за тези приложения.

Функция connection callback		
Активиране на входна точка от VCI	**techComplCode**	**techComplText**
VCI_Dispatch (грешка при четене на съобщение, изпратено от GATE)	ELB_TECH_TECHSRVC_NOT_ AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА ТЕХНИЧЕСКИТЕ УСЛУГИ
VCI_Dispatch (известие от Борсова услуга)	ELB_TECH_XERVICE_NOT_ AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА БОРСОВА УСЛУГА
	ELB_TECH_XERVICE_ AVAILABLE	БОРСОВАТА УСЛУГА Е РАЗПОЛАГАЕМА
	ELB_TECH_NONTRANSPARENT_ FAILOVER	НЕПРОЗРАЧНО ПРЕКЪСВАНЕ
VCI_Dispatch (състояние disconnected, предизвикано от активиране на	ELB_TECH_TECHSRVC_NOT_ AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА ТЕХНИЧЕСКИТЕ УСЛУГИ

Функция connection callback		
Активиране на входна точка от VCI	**techComplCode**	**techComplText**
функцията за обратно повикване)		
VCI_Dispatch (отговор на заявка към VCI_Logout)	ELB_TECH_LOGGED_OUT	ДОСТЪПЪТ НА ПОТРЕБИТЕЛЯ Е ПРЕКРАТЕН УСПЕШНО
VCI_DICONNECT (синхронна заявка)	ELB_TECH_OK(1)	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_REQ_ UNSUCCESSFUL(2)	НЕУСПЕШНО ОБРАБОТВАНЕ НА ЗАЯВКАТА

(1) Означава успешно прекъсване на връзката.
(2) Означава неуспешен опит за прекъсване на връзката.

Таблица 3.3: Функция connection callback

Функция login callback		
Активиране на входна точка от VCI	**techComplCode**	**techComplText**
VCI_Dispatch (отговор на заявка към VCI_Login)	ELB_TECH_LOGGED_IN(1)	ПОТРЕБИТЕЛЯТ Е ПОЛУЧИЛ ДОСТЪП УСПЕШНО
	ELB_TECH_INTERNAL ERROR(2)	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_REQ_ UNSUCCESSFUL(3)	НЕУСПЕШНО ОБРАБОТВАНЕ НА ЗАЯВКАТА
VCI_Dispatch (грешка при четене на съобщение, изпратено от GATE)	ELB_TECH_TECHSRVC_NOT_ AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА ТЕХНИЧЕСКИТЕ УСЛУГИ
	ELB_TECH_PENDING_REQUEST _DELETED(4)	ИЗТРИТА Е ВИСЯЩА ЗАЯВКА

(1) Означава получаване на отговор "Успешно влизане", съдържащ login ID.
(2) Означава вътрешна грешка при обработката на заявка за влизане.
(3) Не се разрешава влизане.
(4) Означава непотвърдена заявка за влизане.

Таблица 3.4: Функция login callback

Функция login callback (при излизане)		
Активиране на входна точка от VCI	**techComplCode**	**techComplText**
VCI_Dispatch (известие от Борсова услуга)	ELB_TECH_XERVICE_NOT_ AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА БОРСОВА УСЛУГА
	ELB_TECH_NONTRANSPARENT_ FAILOVER	НЕПРОЗРАЧНО ПРЕКЪСВАНЕ
	ELB_TECH_PENDING_REQUEST _DELETED(1)	ИЗТРИТА Е ВИСЯЩА ЗАЯВКА
VCI_Dispatch (отговор на заявка към VCI_Logout)	ELB_TECH_LOGGED_OUT(2)	ДОСТЪПЪТ НА ПОТРЕБИТЕЛЯ Е ПРЕКРАТЕН УСПЕШНО
	ELB_TECH_TECHSRVC_NOT_ AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА ТЕХНИЧЕСКИТЕ УСЛУГИ
VCI_Disconnect	ELB_TECH_XERVICE_NOT_ AVAILABLE	НЕРАЗПОЛАГАЕМОСТ НА БОРСОВА УСЛУГА
	ELB_TECH_PENDING_REQUEST _DELETED(1)	ИЗТРИТА Е ВИСЯЩА ЗАЯВКА

(1) Означава непотвърдена заявка за влизане/излизане.
(2) Означава получаване на отговор "Успешно излизане".

Таблица 3.5: Функция login callback (при излизане)

Функция subscription callback		
Активиране на входна точка от VCI	**techComplCode**	**techComplText**
VCI_Dispatch (отговор на заявка към VCI_Subscription)	ELB_TECH_SUBSCRIBED(1)	ПОТРЕБИТЕЛЯТ Е АБОНИРАН ЗА ПОТОКА
	ELB_TECH_INTERNAL ERROR(2)	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
	ELB_TECH_OK(3)	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_RQ_UNSUCCESSUL	НЕУСПЕШНО ОБРАБОТВАНЕ НА ЗАЯВКАТА
VCI_Dispatch (получаване на излъчвания)	ELB_TECH_OK(3)	УСПЕШНО ИЗПЪЛНЕНИЕ

Функция subscription callback		
Активиране на входна точка от VCI	**techComplCode**	**techComplText**
VCI_Dispatch (грешка при четене на съобщение, изпратено от GATE)	ELB_TECH_SUBSCRIPTION_ DELETED	АБОНАМЕНТЪТ Е ИЗТРИТ
	ELB_TECH_PENDING_REQUEST _DELETED(4)	ИЗТРИТА Е ВИСЯЩА ЗАЯВКА
VCI_Dispatch (отговор на заявка към VCI_Unsubscribe)	ELB_TECH_UNSUBSCRIBED(5)	АБОНАМЕНТЪТ ЗА ПОТОКА Е ПРЕКРАТЕН
	ELB_TECH_INTERNAL ERROR(2)	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
VCI_Dispatch (отговор на заявка към VCI_Logout, само за Xervices с Boradcast Extension, виж *точка 2.5.2*)	ELB_TECH_SUBSCRIPTION_ DELETED	АБОНАМЕНТЪТ Е ИЗТРИТ
VCI_Disconnect	ELB_TECH_SUBSCRIPTION_ DELETED	АБОНАМЕНТЪТ Е ИЗТРИТ
	ELB_TECH_PENDING_REQUEST _DELETED(4)	ИЗТРИТА Е ВИСЯЩА ЗАЯВКА

(1) Означава получаване на отговор "Успешно абониране".
(2) Означава вътрешна грешка при обработката на заявка за абониране.
(3) Означава успешно получаване на излъчена емисия.
(4) Означава непотвърдена заявка за абониране/прекратяване на абонамент.
(5) Означава получаване на съобщение "Успешно прекратяване на абонамент".

Таблица 3.6: Функция subscription callback

Функция submit callback		
Активиране на входна точка от VCI	**techComplCode**	**techComplText**
VCI_Dispatch (отговор на заявка към VCI_Submit)	ELB_TECH_OK	УСПЕШНО ИЗПЪЛНЕНИЕ
	ELB_TECH_REQ_UNSUCCESSUL	НЕУСПЕШНО ОБРАБОТВАНЕ НА ЗАЯВКАТА
	ELB_TECH_INTERNAL ERROR	ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА
VCI_Dispatch (грешка при четене на съобщение, изпратено от GATE)	ELB_TECH_PENDING_REQUEST_DELETED	ИЗТРИТА Е ВИСЯЩА ЗАЯВКА
VCI_Dispatch (известие от Борсова услуга)	ELB_TECH_PENDING_REQUEST_DELETED	ИЗТРИТА Е ВИСЯЩА ЗАЯВКА
VCI_Dispatch (отговор на заявка към VCI_Logout)	ELB_TECH_PENDING_REQUEST_DELETED	ИЗТРИТА Е ВИСЯЩА ЗАЯВКА
VCI_Disconnect	ELB_TECH_PENDING_REQUEST_DELETED	ИЗТРИТА Е ВИСЯЩА ЗАЯВКА

Таблица 3.7: Функция submit callback

4 VALUES API. Примери за използване

4.1 Обзор

В този раздел са представени избрани извадки от кода, които илюстрират как се използват входните точки от интерфейса за повикване на VALUES API. Всеки пример за програмиране включва извадки от кода и обяснителен текст. Извадките от кода включват повиквания към VALUES API, спесификация на параметрите, заделяне на памет и примери на функциите за обратно повикване на приложението.

Разделът е организиран в отделни точки. Във всяка точка е описана пътеката от иницииране на сесия, влизане в Борсови приложения, обработка на приложни заявки и отговори на приложни заявки (включително диспечерския цикъл), излизане от Борсовото приложение и прекратяване на сесията.

Дадените в този раздел извадки от кода имат за цел да илюстрират как някои задачи, които са свързани с програмирането на VALUES, могат да бъдат реализирани в потребителското приложение. Те не представляват завършена примерна програма. Не се препоръчва използването им като шаблони за базирани на VALUES клиентски приложения без нужните адаптации.

За целите на примерите се приема, че се използва платформата Sun Solaris.

4.2 Иницииране на сесия във VALUES

За установяване на връзка между потребителското приложение и VALUES трябва да се използва входната точка VCI_Connect. След успешно изпълнение на VCI_Connect, потребителското приложение може да се абонира за потоци от данни и да получава излъчвания на данни. Освен това, то може да установява връзка с различните Борсови услуги, да изпраща приложни заявки и да получава съответните отговори.

Следният списък показва как потребителското приложение може да заделя и освобождава памет при използването на VCI_Connect:

- създава се connection handler (оператор/маркер на връзката)
- заделя се памет за данни за заявката
- заделя се памет за контекстуални данни
- повиква се VCI_Connect (синхронно)
- в зависимост от techComplCode на VCI_Connect
- ако ELB_TECH_OK *(connected)*
- освобождава се паметта за данни за заявката
- или *(при неуспешен опит за свързване)*

- освобождава се паметта за данни за заявката
- освобождава се паметта за контекстуални данни
- изтрива се connection handler

- изчаква се известие за промяна на състоянието чрез connection callback
- в зависимост от techComplCode на connection callback
- ако ELB_TECH_TECHSRVC_NOT_AVAILABLE (*disconnected*)
- освобождава се паметта за контекстуални данни
- изтрива се connection handler

- при други кодове за изпълнение – няма други действия

- повиква се VCI_Disconnect (синхронно)
- ако кодът за изпълнение на VCI_Disconnect не е нито ELB_TECH_INVALID_PARAMETER, нито ELB_TECH_NOT_CONNECTED (*disconnected*)
- освобождава се паметта за контекстуални данни
- изтрива се connection handler

Примерен код на VCI_Connect:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <poll.h>                    // Тази библиотека е налична на платформата Sun.
                                     // Функцията за прослушване тук се използва за прослушване
                                     // на VMQ (VALUES Message Queue).
// специфични файлове за VALUES
#include "Values.h"                  // Този файл декларира всички типове и функции на
                                     // на VCI (VALUES Call Interface).
#include "elbcodetech.h"             // Този файл декларира кодовете за техническо изпълнение.

// файлове за Eurex
#include "XEUR_elbcode.h"            // Този файл декларира кодовете за функционално изпълнение.
#include "DRIV_data_types.hxx"       // Този файл съдържа константите, които определят големините
                                     // и валидните стойности на полетата с данни.
#include "DRIV_app_rid.h"            // Този файл съдържа идентификатори на приложните заявки
#include "DRIV_login.h"              // Този файл съдържа дефиниция на структурата.
                                     // на оторизационните данни за Eurex.
#include "EntSLegOrdr.hxx"           // Включва структурни дефиниции, необходими
                                     // за подаване на приложната заявка Enter Order по-долу.
// файлове за Xetra
```

```
#include "app_rid.h"
#include "xbrdcast.h"
#include "subjectXetra.h"
#include "xetra_login.h"
#include "vld_val.h"
// глобални структури и променливи
// повечето от тях се използват за комуникация на функциите за обратно повикване с основната
// програма или за да се гарантира, че променливите, които трябва да са достъпни както във
// функциите за обратно повикване, така и в основната програма, са в обсега и на двете среди.
// Реалните приложения би трябвало да използват структури за индивидуални контекстуални данни
// и malloc()'d за заделяне на памет за тази обработка.
char myContextData [100];          // connectCallback получава указател към
                                   // myContextData в callbackCntextData
                                   // от VALUES и трябва да реализира достъп до тези данни.
                                   // Данните трябва да са валидни, докато сесията
                                   // бъде прекратена.

LoginReqDataT*   LoginData;        // блок от данни за заявка login
SubmitReqDataT* SubmitData;        // блок от данни за заявка submit

optEntSLegOrdrRequestT     my_appl_request_struct;
                                   // responseCallback
                                   // получава указател appReq =
                                   // &my_appl_request_struct, за да
                                   // идентифицира заявката, която е задействала отговора.
                                   // Следователно, my_appl_request_struct
                                   // трябва да се запише, примерно като се декларира като
                                   // глобална променлива. Данните трябва да са валидни, докато
                                   // отговорът бъде обработен.

int callbackLoginId;               // получава loginID в рамките на функцията loginCallback
int subsId;                        // идентификатор на абонамента за излъчвания на данни (subscription ID)

#define XERVICE_YET_UNKNOWN (–1)
int xetraDbApplId = XERVICE_YET_UNKNOWN;   // application ID на Xervice за транзакции на Xetra
                                           // да се запише от Connect callback
int eurexDbApplId = XERVICE_YET_UNKNOWN;   // application ID на Xervice за транзакции на Eurex
                                           // да се запише от Connect callback

int main (int argc, char** argv)
{
        // Локални променливи
        int    fd;                 // fd файловият дескриптор наVMQ.
        char  prodMode;            // Приложението може да използва prodMode, за да
                                   // информира потребителя за вида на експлоатационния режим.

    // Дефинират се променливи във VCI формат:
    ReqCntrlT           reqControl1;        // записи за контрол на заявките,
    ReqCntrlT           reqControl2;        // за всеки потребител
    CnctReqDataT*       cnctreqData;        // блок данни за заявки connection
    SubmitReqDataT*      submitreqData;     // блок данни за заявки submit
    DiscnctReqDataT*    discnctreqData;     // блок данни за заявки disconnect
    CnctRespDataT       cnctrespData;       // блок данни за отговори на заявки connection
    StatusDataT         statusData;         // запис на статуса на функц. изпълнение - VCI
    AppCntxtDataT*      callbackContextData;// блок за контекстуални данни на application callback
```

```
    DRIVLoginAuthorizationDataT*  DRIV_AuthData;        // Структура на оторизационните данни за Eurex
    XetraLoginAuthorizationDataT*  xetraAuthData;        // Структура на оторизационните данни за Xetra

    strcpy(reqControl1.VCIver, CVN_012);   // Посочва се версия на VALUES с цел проверка за съвместимост;

    // заделя се памет за cnctreqData (тази памет трябва да е free()d след
    // обръщане от VCI_Connect)
    cnctreqData = (CnctReqDataT*) malloc( sizeof(CnctReqDataT) );

    // заделя се памет за контекстуални данни за използване през цялата сесия във VALUES
    callbackContextData = (AppCntxtDataT*) malloc( sizeof(AppCntxtDataT ));

    // попълват се полетата на структурата cnctreqData
    // userId и паролата идват от потребителя или от файл, не се фиксират в кода
    strcpy(cnctreqData->userID, "myuser");              //  userID в операционната система на FE-
                                                        //  или БД-приложение за VALUES
    strcpy(cnctreqData->password, "mypassword");        //  парола в операционната система на FE за userID

    // попълват се полетата на структурата callbackContextData
    // В този пример не са използвани контекстуални данни
    callbackContextData->custBlockSize = 0;
    callbackContextData->custData = NULL;

    // Повиква се VCI_Connect, за да свърже приложението с VALUES.
    // Това е необходимо за влизане в Back End системите.
    VCI_Connect( &reqControl1,
    cnctreqData,
    connectCallback,
    callbackContextData,
    &cnctrespData,
    &statusData
    );

    //Оценява се дали повикването е било успешно или не е успешно.
    ret_status_handling( &statusData );

    ////////////////////////////////////////////////////////////
    // Имаме резултатни данни, получени от VCI_Connect.
    // Сега записваме съответните данни за по-нататъшните обработки във VALUES.
    ////////////////////////////////////////////////////////////

    // Видът на експлоатационният режим е записан.
    prodMode = cnctrespData.prodMode;
    // Файловият дескриптор на VMQ е записан.
    fd = atoi(cnctrespData.VMQname);
    // connection ID е записан, за да идентифицира сесията
    // при всички следващи повиквания на интерфейса.
    reqControl1.connectionID = cnctrespData.connectionID;

    // Със следващата функция прослушваме VMQ и извикваме VCI_Dispatch при настъпване на събитие.
    // Това повикване се извършва за получаване на асинхронно изпратени данни
    // за разполагаемостта на Xervices
    poll_and_dispatch ( &reqControl1, fd );
    // Оттук продължава специфичната обработка в приложението.
}
```

4.3 Диспечиране на приложението след известяване за събитие

Входната точка VCI_Dispatch се използва от потребителското приложение за получаване на отговори, излъчвания и изключения. VCI_Dispatch прочита данните за отговора от VMQ и ги предава на съответна функция за обратно повикване.

Примерен код за прослушване на VMQ и повикване на VCI_Dispatch:

```
void poll_and_dispatch (        ReqCntrlT       *reqControl,    // запис за контрол на заявката
                                int             fd              // файлов дескриптор на VMQ
                                )
{
        StatusDataT  statusData;        // статус на функционално изпълнение, VCI
        struct pollfd  pfd;             // Тази променлива се използва за записване на резултата
                                        // от прослушването на VMQ.

    // Прослушване на VMQ.
    // Важно е да НЕ СЕ чете от VMQ.
    // Самото четене се извършва от VCI_Dispatch.
    pfd.fd = fd;
    pfd.events = 0;
    pfd.events = pfd.events | POLLIN;

    pfd.events   = pfd.events   | POLLRDNORM;
    pfd.events   = pfd.events   | POLLRDBAND;
    pfd.events   = pfd.events   | POLLPRI;
    pfd.events   = pfd.events   | POLLERR;

    ако (poll(&pfd, 1, -1) < 0)             // Блокира до настъпване на събитие във VMQ.
                                            // Използвай  select() за други платформи.
    {
            printf("\n Poll failed");
            break;
    }

    / Ако има съобщение, повиква се VCI_Dispatch.
    ако ( (pfd.revents & POLLERR) != 1 )
    {
            // Повиква се VCI_Dispatch за прочитане на данни за отговора и VCI_Dispatch при събитие.
            VCI_Dispatch (          reqControl,
                                    &statusData
                                    );
            // Оценява се дали повикването е било успешно или неуспешно.
            ret_status_handling(&statusData);
    }
    или
    {
            printf("\n Грешка в асинхронния канал");
    }
    return;
}
```

4.4 Получаване на събития при свързване

Следната извадка от кода илюстрира как в потребителското приложение може да бъде реализирана функция за обратно повикване, която да извършва асинхронна обработка на събитията при свързване.

Примерен код на connection callback:

```
void connectCallback (      ReqCntrlT *reqControl,
                            CallBkAppDataT*appData,
                            AppCntxtDataT*callBackCntxtData,
                            StatusDataT *statusDataGlobal
                            )
{
      //използва се за декодиране на съобщенията "Xervice Available" и /"Xervice not available"
      XerviceInfoT* pxinfo;
      printf( "\n***************************************** \n");
      printf( "\tConnection Callback \n");
      printf( "***************************************** \n");
      switch( statusDataGlobal->techComplCode )
    {
      // Тази функция не обработва всички кодове за изпълнение, които могат да се получат.
      case ELB_TECH_OK:
            printf( "Заявката за изключване е изпълнена успешно\n" );
             break;
      case ELB_TECH_REQ_UNSUCCESSFUL:
             // При това изключение приложението е длъжно да реши,
             // въз основа на детайлите за контекста
             // (вид изключение, данни за заявката, търговска фаза и др.), дали заявката
             // трябва да бъде подадена повторно или не трябва.
             printf( "\n%s\n", statusDataGlobal->techComplText );
             // Кодът за функционално изпълнение предоставя детайлна информация:
             printf( "%s\n", statusDataGlobal->complText );
             break;
      case ELB_TECH_LOGGED_OUT:
             // приложението е излязло от съответната Xervice.
             // Обикновено това не изисква други действия от
             // connect callback, тъй като бива активиран и съответния login callback.
             break;
      case ELB_TECH_XERVICE_AVAILABLE:
             // Извличат се идентификаторите (Xervice IDs) на интересуващите ни Xervices
             // За да ползва възможността за работа на множество борси, приложението трябва да записва
             // всеки Xervice Id, постъпващ с Xervice Class, който приложението може да обработва,
             // както и допълнителна информация от  XerviceInfoT. Примерният код
             // не работи с множество борси, а само предварително определена борса.
             pxinfo = (XerviceInfoT*) appData->appRespData;
             // Интересува ни конкретно dbApplID на Transaction Xervice (услуга за транзакции) предлагана
             // от Xetra Frankfurt (MIC е “XETR”)
             ако    (pxinfo->applClass == XETRA_TXN_XCLASS &&
                    memcmp(pxinfo->exchApplId, “X” EXCH_ID_COD_XETRA, 4) == 0)
             {
                    xetraDbApplId = reqControl->dbApplID;
                    break;
             }
```

```
                // по същия начин за Eurex
                ако     (pxinfo->applClass == EUREX_TXN_XCLASS &&
                        memcmp(pxinfo->exchApplId, EXCH_APPL_ID_EUREX, 4) == 0)
                {
                        eurexDbApplId = reqControl->dbApplID;
                }
                break;
        case ELB_TECH_XERVICE_NOT_AVAILABLE:
                // при стартиране на приложението, това събитие се изпраща на всяка Xervice, която е
                // конфигурирана върху  MISS, но не е разполагаема в момента. По този начин може да
                // получаваме XerviceInfo и оттук.
                pxinfo = (XerviceInfoT*) appData->appRespData;
                // Интересува ни конкретно  dbApplID на Transaction Xervice (услуга за транзакции) предлагана
                // от Xetra Frankfurt (MIC е “XETR”)
                ако     (pxinfo->applClass == XETRA_TXN_XCLASS &&
                        memcmp(pxinfo->exchApplId, “X” EXCH_ID_COD_XETRA, 4) == 0)
                {
                                xetraDbApplId = reqControl->dbApplID;
                                break;
                }
                // по същия начин за Eurex
                ако     ( pxinfo->applClass == EUREX_TXN_XCLASS &&
                        memcmp(pxinfo->exchApplId, EXCH_APPL_ID_EUREX, 4) == 0)
                {
                        eurexDbApplId = reqControl->dbApplID;
                }
                break;
                // Приложението може да изчака, докато борсата стане разполагаема.
        case ELB_TECH_INTERNAL_ERROR:
                // Приложението трябва да се изключи и да направи опит за рестартиране. Тази грешка най-често
                // се дължи на проблеми с конфигурацията или настройката на MISS или работната станция.
        case ELB_TECH_TECHSRVC_NOT_AVAILABLE:
                // В този момент връзка на приложението с VALUES ще бъде прекъсната.
                // То може да са свърже отново
                // след рестартиране на техническите услуги (ако необходимо от системния оператор)
                printf( "\n%s\n", statusDataGlobal->techComplText );
                break;
        default:
                printf( "\nUnknown Error! Connection Id: %d\n", reqControl->connectionID );
                if (callBackCntxtData)
                {
                        printf("callBackCntxtData.custData = %s\n", (char*)callBackCntxtData->custData);
                }
                printf("Application Version = %d\n", appData->applVersion);
                break;
    }
    return;

}
```

4.5 Осъществяване на достъп до Борсовите услуги

За осъществяване на достъп до Борсовите услуги трябва са използва входната точка VCI_Login. Преди да изпраща приложни заявки, потребителят трябва да получи оторизация от съответната Борсова услуга. Тези Xervices, които реализират Broadcast Extension (виж *точка 2.5.2*) изискват оторизация и при абониране за излъчвания. Множество потребители могат да работят с дадена Борсова услуга, използвайки една и съща сесия във VALUES.

Следният списък показва как потребителското приложение може да заделя и освобождава памет при използването на VCI_Login и VCI_Logout:

- създава се login handler (оператор/маркер на влизането)
- заделя се памет за данни за заявката
- заделя се памет за контекстуалните данни
- login handler се маркира като "pending"
- повиква се VCI_Login (асинхронно)
- ако techComplCode за VCI_Login не е ELB_TECH_OK *(т.е. ако опитът за влизане е неуспешен*)
- освобождава се паметта за контекстуалните данни
- освобождава се паметта за данни за заявката
- изтрива се login handler
- изчаква се известие за промяна на състоянието чрез login callback
- в зависимост от techComplCode на login callback:
- ако е ELB_TECH_LOGGED_IN *(logged in)*
- login handler се маркира като "logged in"
- освобождава се паметта за данни за заявката
- ако е ELB_TECH_UNSUCCESSFULL или ELB_TECH_INTERNAL_ERROR *(грешка при обработката на заявката за влизане)*
- освобождава се паметта за контекстуалните данни
- освобождава се паметта за данни за заявката

- изтрива се login handler
- ако е ELB_TECH_NON_TRANSPARENT_FAILOVER
- няма други действия
- ако е ELB_TECH_LOGGED_OUT или ELB_TECH_TECHSRVC_NOT_AVAILABLE *(logged out)*
- освобождава се паметта за контекстуалните данни
- изтрива се login handler
- ако е ELB_TECH_PENDING_REQUEST_DELETED
- ако login handler е "pending" *(грешка при обработката на заявката за влизане)*
- освобождава се паметта за контекстуалните данни
- освобождава се паметта за данни за заявката
- изтрива се login handler
- или *(грешка при обработката на заявката за излизане)*
- няма други действия
- ако е ELB_TECH_XERVICE_NOT_AVAILABLE
- ако login handler е "logged in" *(logged out)*
- освобождава се паметта за контекстуалните данни
- изтрива се login handler
- или *(грешка при обработката на заявката за влизане)*
- освобождава се паметта за контекстуалните данни
- освобождава се паметта за данни за заявката
- изтрива се login handler
- повиква се VCI_Logout (асинхронно)
- няма други действия

Примерен код на VCI_Login:

```
// Дублира се записа за контрол на заявките с цел използването му за друг потребител.
reqControl2 = reqControl1;

// Заделя се памет за LoginData
//(тази памет трябва да е свободна в login/out callback).
LoginData = (LoginReqDataT*) malloc( sizeof(LoginReqDataT) );

// Попълват се полетата на структурата  reqData за първия потребител.
// Задава се версия на приложение, както е дефинирана в DRIV_data_types.hxx.
LoginData->applVersion = XEUR_AVN_090

// userID и паролата идват от потребителя или от файл и не трябва да са
// фиксирани в кода, както са тук.
memcpy(LoginData->userID, "MEMBRTRD001", LOGIN_MAX_USERID);
DRIV_AuthData = (DRIVLoginAuthorizationDataT*) malloc(sizeof(DRIVLoginAuthorizationDataT));
memcpy(DRIV_AuthData->password, "MYPASSWD", DRIV_LOGIN_MAX_PWDID);  //парола за userID

// Тук трябва да се използва зададения от борсата application ID.
memcpy(DRIV_AuthData->applicationId, "EUREXXXXXXXXXXX", DRIV_LOGIN_APPLICATION_ID );
LoginData->authorizationData          = (void*) DRIV_AuthData;
LoginData->authorizationDataLength  = sizeof(DRIVLoginAuthorizationDataT);

strcpy(reqControl1.appDescr, "Eurex");          // Това е незадължително описание.
// проверява се дали Xervice е обявена като разполагаема: (това става вътре в connectCallback)
assert( eurexDbApplId != XERVICE_YET_UNKNOWN );
reqControl1.dbApplID = eurexDbApplId;       // Ще се извърши влизане в Eurex.

// Извиква се VCI_Login за първия потребител. С това потребителят получава оторизация
// от борсовия Back End за подаване на приложни заявки.
VCI_Login(    &reqControl1,
              LoginData,
              loginCallback,
              NULL,
              NULL,
              &statusData
              );
// Оценява се дали повикването е било успешно или неуспешно.
ret_status_handling(&statusData);

// В тази функция прослушваме VMQ и при събитие повикваме VCI_Dispatch.
poll_and_dispatch ( &reqControl1, fd );

// loginID е получен в loginCallback.
reqControl1.loginID = callbackLoginId;

// Заделя се памет за LoginData
// (тази памет трябва да бъде свободна в login/out callback).
LoginData = (LoginReqDataT*) malloc(sizeof(LoginReqDataT));
// Попълват се полетата на структурата reqData за втория потребител.
// userID и паролата идват от потребителя или от файл и не трябва да са
// фиксирани в кода, както са тук.
memcpy(LoginData->userID, "MEMBRTRD002", LOGIN_MAX_USERID);
xetraAuthData = (XetraLoginAuthorizationDataT*) malloc(sizeof(XetraLoginAuthorizationDataT));
```

```
memcpy(xetraAuthData->password, "MYPASSWD", XETRA_LOGIN_MAX_PWDID);      // парола за userID
LoginData->authorizationData = (void*) xetraAuthData;
LoginData->authorizationDataLength = sizeof(XetraLoginAuthorizationDataT);

strcpy(reqControl2.appDescr, "Xetra");          // Това е незадължително описание.
// проверява се дали Xervice е обявена като разполагаема:
assert( xetraDbApplId != XERVICE_YET_UNKNOWN );
reqControl2.dbApplID = xetraDbApplId; // Ще се извърши влизане в Xetra.
// Повиква се VCI_Login втория потребител.
VCI_Login(    &reqControl2,
              LoginData,
              loginCallback,
              NULL,
              NULL,
                    &statusData
                    );

// Оценява се дали повикването е било успешно или неуспешно.
ret_status_handling(&statusData);

// В тази функция прослушваме VMQ и при събитие повикваме VCI_Dispatch.
poll_and_dispatch ( &reqControl2, fd );

// loginID е получен от loginCallback.
reqControl2.loginID = callbackLoginId;
```

4.6 Получаване отговори на заявки за влизане

Следната извадка от кода илюстрира как в потребителското приложение може да бъде реализирана функция за обратно повикване, която да обработва асинхронно отговори на заявки за влизане, подадени от потребителя.

Примерен код на login callback:

```
extern LoginReqDataT*     LoginData;              // /блок от данни за login request
void loginCallback (      ReqCntrlT          *reqControl,
                          CallBkAppDataT     *appData,
                          AppCntxtDataT      *callBackCntxtData,
                          StatusDataT        *statusDataGlobal
                          )
{
        // указателят  void се прехвърля в реалния му тип
        LoginRespDataT *respData = (LoginRespDataT*) appData->appRespData;
        printf( "\n************************************** \n");
        printf( "\tLogin/out Callback \n");
        printf( "************************************** \n");
        switch( statusDataGlobal->techComplCode )
{
                // Тук са само някои от обработваните кодове за изпълнение в elbcodetech.h.
                case ELB_TECH_LOGGED_IN:
                        printf( "Logged In" );
                        printf( "\n LOGIN ID: %ld\n", respData->loginID );
                        // току-що полученият loginID се записва в глобалната променлива
```

```
                callbackLoginId = respData->loginID;
                break;
        case ELB_TECH_LOGGED_OUT:
                free(LoginData);
                printf( "Logged Out" );
                break;
        case ELB_TECH_REQ_UNSUCCESSFUL:
                printf( "\n%s\n", statusDataGlobal->techComplText );
                // Кодът за функционално изпълнение предоставя детайлна информация:
                printf( "%s\n", statusDataGlobal->complText );
                break;
        case ELB_TECH_XERVICE_AVAILABLE:
        case ELB_TECH_XERVICE_NOT_AVAILABLE:
        case ELB_TECH_TECHSRVC_NOT_AVAILABLE:
        case ELB_TECH_NONTRANSPARENT_FAILOVER:
                printf( "\n%s\n", statusDataGlobal->techComplText );
                break;
        default:
                printf( "\n%s\n", statusDataGlobal->techComplText );
                printf( "Connection Id: %d\n", reqControl->connectionID );
        ако (callBackCntxtData)
        {
                printf("callBackCntxtData.custData = %s\n", (char*)callBackCntxtData->custData);
        }
        break;
    }
    return;
}
```

4.7 Подаване на приложни заявки

За да изпраща заявки за обработки от едно или друго Борсово приложение (приложни заявки), потребителското приложение трябва да използва входната точка VCI_Submit. Потребителят трябва да спесифицира код на заявката, данни за заявката и функция за получаване обратни повиквания.

Следният списък показва как потребителското приложение може да заделя и освобождава памет при използването на VCI_Submit:

- създава се submission handler (оператор/маркер на подаванията на заявки)
- заделя се памет за контекстуални данни
- заделя се памет за данни за заявката
- повиква се VCI_Submit (асинхронно)
- ако techComplCode на VCI_Submit не е ELB_TECH_OK *(т.е. ако опитът за подаване на заявка е неуспешен)*
- освобождава се паметта за контекстуални данни

- освобождава се паметта за данни за заявката
- изтрива се submission handler

- изчаква се известие за промяна на състоянието чрез submission callback
- в зависимост от techComplCode на submission callback
  - за всички кодове за изпълнение
    - освобождава се паметта за контекстуални данни
    - освобождава се паметта за данни за заявката
    - изтрива се submission handler

Примерен код на VCI_Submit:

```
// В този момент трябва да се попълнят полетата от структурата на приложната заявка.
// В общия случай тези полета не са фиксирани в кода, а се попълват от потребителя.
// Това е пример за въвеждане на поръчка за опция върху акции в системата на Eurex.
memcpy(my_appl_request_struct.header.exchApplId, EXCH_APPL_ID_EUREX, EXCH_APPL_ID_LEN);
memcpy(my_appl_request_struct.header.prodLine, PROD_LINE_OPTION, PROD_LINE_LEN)
memset(my_appl_request_struct.header.membExchIdCodOboMS, EXCH_CONST_SPACE,
        MEMB_EXCH_ID_COD_OBO_MS_LEN);
my_appl_request_struct.extension.acctTypCod = ACCT_TYP_COD_AGENT;
my_appl_request_struct.extension.acctTypNo   = ACCT_TYP_NO_ONE;
memcpy(my_appl_request_struct.extension.txtGrp.cust, "My Customer ", CUST_LEN);
memcpy(my_appl_request_struct.extension.txtGrp.userOrdrNum, "123456789012",
        USER_ORDR_NUM_LEN);
memcpy(my_appl_request_struct.extension.txtGrp.text, "FreeFormText", TEXT_LEN);
memcpy(my_appl_request_struct.extension.membClgIdCod, "CLGMB", MEMB_CLG_ID_COD_LEN);
my_appl_request_struct.extension.prcRsblChkInd = EXCH_CONST_NO;

my_appl_request_struct.basic.buyCod = EXCH_CONST_BUY;
memcpy(my_appl_request_struct.basic.optCntrIdGrp.prodId, "ODAX", PROD_ID_LEN);
my_appl_request_struct.basic.optCntrIdGrp.cntrClasCod = CNTR_CLAS_COD_CALL;
memcpy(my_appl_request_struct.basic.optCntrIdGrp.cntrExpMthDat, "06", CNTR_EXP_MTH_DAT_LEN);
memcpy(my_appl_request_struct.basic.optCntrIdGrp.cntrExpYrDat, "2006", CNTR_EXP_YR_DAT_LEN);
memcpy(my_appl_request_struct.basic.optCntrIdGrp.cntrExerPrc, "0004550", CNTR_EXER_PRC_LEN);
my_appl_request_struct.basic.optCntrIdGrp.cntrVersNo = CNTR_VERS_NO_ZERO;
memset(my_appl_request_struct.basic.trdrIdGrp.partSubGrpCod, EXCH_CONST_SPACE,
        PART_SUB_GRP_COD_LEN);
memset(my_appl_request_struct.basic.trdrIdGrp.partNo, EXCH_CONST_SPACE, PART_NO_LEN);
memcpy(my_appl_request_struct.basic.ordrQty, "+000000000177", DRIV_ORDR_QTY_LEN);
memcpy(my_appl_request_struct.basic.ordrExePrc, "+0000000002250", DRIV_ORDR_EXE_PRC_LEN);
my_appl_request_struct.basic.ordrResCod = EXCH_CONST_SPACE;
memcpy(my_appl_request_struct.basic.ordrExpDat, "20060630", ORDR_EXP_DAT_LEN);
my_appl_request_struct.basic.opnClsCod = OPN_CLS_COD_OPEN;

// Задава се вида на заявката. Дефиницията е във файл "DRIV_app_rid.h".
reqControl1.reqID = DRIV_ENTER_SINGLE_LEG_ORDER_RID;
```

```
// заделя се памет за callbackContextData
callbackContextData = (AppCntxtDataT*) malloc(sizeof(AppCntxtDataT));

// Със SubmitData (global) реално предаваме данните на приложната заявка
// на VALUES Call Interface.
SubmitData.appReq = &my_appl_request_struct;
                                    // Този указател се връща на
                                    // съответния application response callback
                                    // и може да се използва за идентифициране на заявката,
                                    // която е активирала response callback
SubmitData.appReqBlockSize = sizeof(my_appl_request_struct);

// Попълват се някои индивидуални данни.
strcpy(myContextData, "това е низ от контекстуални данни");     // Контекстуалните данни се записват,
                                                                // за да са достъпни в response callback.
// индивидуализира се callbackContextData
callbackContextData->custData = myContextData;                  // Указателят към контекстуалните данни се
                                                                // предава на VALUES. Той се връща на
                                                                // приложението в application callback.
callbackContextData->custBlockSize = sizeof(myContextData);

// Повиква се VCI_Submit за първия потребител. С това приложната заявка реално се предава
// на Back-End приложението.
VCI_Submit(     &reqControl1,
                &SubmitData,
                responseCallback,
                callbackContextData,
                &statusData
                );

// Оценява се дали повикването е било успешно или неуспешно.
ret_status_handling(&statusData);

// В тази функция прослушваме VMQ и при събитие повикваме VCI_Dispatch.
poll_and_dispatch ( &reqControl1, fd );
```

4.8 Получаване на отговори на приложни заявки

Следната извадка от кода илюстрира как в потребителското приложение може да бъде реализирана функция за обратно повикване, която да извършва асинхронна обработка на отговори на приложни заявки.

Примерен код на application request callback:

```
void responseCallback (     ReqCntrlT           *reqControl,
                            CallBkAppDataT      *appData,
                            AppCntxtDataT       *appCntxtData,
                            StatusDataT         *statusDataGlobal
                            )
{
```

```
int                             responseSize;
optEntSLegOrdrRequestT      *my_appl_req_struct;
optEntSLegOrdrResponseT     *responseData;

printf( "\n************************************** \n");
printf( "\tResponse Callback \n");
printf( "************************************** \n");

switch( statusDataGlobal->techComplCode )
{
// Тази функция не обработва всички възможни кодове за изпълнение, които могат да бъдат получени
 case ELB_TECH_OK:

        // Структурата  reqControl съдържа reqID на заявката, която активира
        // тази функция за обратно повикване. reqID може да бъде извлечен в този момент,
        // за да определи по-нататъшната обработка.

        // Структурата  appReqData на параметъра appData
        // съдържа заявката, която е била изпратена от приложението
        // с повикването на VCI_Submit.
        // Тази информация може да бъде извлечена както следва:
        my_appl_req_struct = (optEntSLegOrdrRequestT*) appData->appReqData;

        // Структурата appData съдържа данните за приложния отговор
        // и размера на този отговор:
        responseSize = appData->appRespBlockSize;
        responseData = (optEntSLegOrdrResponseT*) appData->appRespData;

        // Структурата appCntxtData съдържа контекстуалните данни, които могат да бъдат
        // извлечени подобно на appReqData или appRespData.
        ////////////////////////////////////////////////////////////////////////////
        // За целите на верификацията се изпращат данните от appCntxtDataT,
        // както и някои данни за заявката и отговора от appData.
        ////////////////////////////////////////////////////////////////////////////
        printf("appCntxtDataT.custData = %s\n", (char*)appCntxtData->custData);
        printf("application_request_structure->basic.optCntrIdGrp.prodId = %*.*s\n",
                PROD_ID_LEN, PROD_ID_LEN,
                my_appl_req_struct->basic.optCntrIdGrp.prodId );
         printf("responseData->basic.ordrNo = %*.*s\n",
                DRIV_ORDR_NO_LEN, DRIV_ORDR_NO_LEN,
                responseData->basic.ordrNo );
         break;
case ELB_TECH_REQ_UNSUCCESSFUL:
        // При това изключение приложението е длъжно да реши, въз основа на детайлите за контекста
        // (декодиране на изключението, данни за заявката, търговска фаза и др.), дали заявката
        // трябва да бъде подадена повторно или не трябва.
        printf( "\n%s\n", statusDataGlobal->techComplText );
        // Кодът за функционално изпълнение предоставя детайлна информация:
        printf( "%s\n", statusDataGlobal->complText );
        break;
case ELB_TECH_PENDING_REQUEST_DELETED:
        // Това изключение възниква, примерно, ако приложението излезе от Борсовата услуга
        // при наличие на висящи приложни заявки
```

```
                printf( "\n%s\n", statusDataGlobal->techComplText );
                break;
                default:
                printf( "\n%s\n", statusDataGlobal->techComplText );
                printf( "Connection Id: %d\n", reqControl->connectionID );
                break;

        }

        // освобождаване на appReqData – тук се приема, че данните са били  malloc()ed преди
        // да предаването им на VCI_Submit().
        // Разбира се, ако сте използвали локална структура за функцията,
        // тук тя не трябва да се подлага на free()ed , обаче данните
        // трябва да останат в обхват и да не се променят до получаване на response callback
        // (често пъти malloc е по-удобно).
        free(appData->appReqData);
}
```

4.9 Абониране за поток от данни

За да се абонира за поток от данни, потребителското приложение трябва да използва повикване към VCI_Subscribe. Преди да се пристъпи към абониране, трябва да е започната сесия. Тези Xervices, които реализират Broadcast Extension (виж *точка 2.5.2*), изискват и валиден login. Потребителското приложение трябва да посочи желания поток от данни и функция за обратно повикване.

Следният списък показва как потребителското приложение може да заделя и освобождава памет при използването на VCI_Subscribe и VCI_Unsubscribe:

- създава се subscription handler (оператор/маркер на абонираният)
- заделя се памет за данни за заявката
- заделя се памет за контекстуални данни
- subscription handler се маркира като "pending"
- повиква се VCI_Subscribe (асинхронно)
- ако techComplCode за VCI_Subscribe не е ELB_TECH_OK *(т.е. ако опитът за абониране е неуспешен*)
- освобождава се паметта за контекстуалните данни
- освобождава се паметта за данни за заявката
- изтрива се subscription handler
- изчаква се известие за промяна на състоянието чрез subscription callback

- в зависимост от кода за изпълнение на subscription callback
- ако е ELB_TECH_LOGGED_IN *(broadcast)*
- няма други действия
- ако е ELB_TECH_SUBSCRIBED *(subscribed)*
- subscription handler се маркира като "subscribed"
- ако е ELB_TECH_SUBSCRIPTION_DELETED, ELB_TECH_INTERNAL_ERROR или ELB_TECH_UNSUBSCRIBED *(unsubscribed)*
- освобождава се паметта за контекстуалните данни
- освобождава се паметта за данни за заявката
- изтрива се subscription handler
- ако е ELB_TECH_PENDING_REQUEST_DELETED
- ако subscription handler е "pending" *(грешка при обработката на заявката за абониране)*
- освобождава се паметта за контекстуалните данни
- освобождава се паметта за данни за заявката
- изтрива се subscription handler
- или *(грешка при обработката на заявката за прекратяване на абонирането)*
- няма други действия
- повиква се VCI_Unsubscribe
- няма други действия

Примерен код на VCI_Subscribe:

```
void subscribe_function (void)
{
    // попълват се параметрите на reqControl
    strcpy(reqControl.appDescr, "Xetra");                    // Това е незадължително описание.
    assert(xetraDbApplId != XERVICE_YET_UNKNOWN); // вече трябва да е обявено от connect callback
    reqControl.dbApplID = xetraDbApplId;                     // Ще се извърши абониране за Xetra.
    strcpy(reqControl.VCIver, CVN_012);                      // Посочва се версия на VALUES с цел проверка за
                                                             // съвместимост;
                                                             // VCI_VERSION е константа във VALUES.
```

```
    // В този момент connection ID, получен като върнат  параметър
    // от VCI_Connect, се използва за идентифициране на сесията.
    reqControl.connectionID = myConnectionID;
    // За Xervice с Broadcast Extension се изисква в полето loginID да е зададен
    // валиден ID, получен чрез login callback, а не просто нула като тук.
    reqControl.loginId = 0;

    // Заделя се памет за reqData (тази памет трябва да бъде освободена след повикване на VCI_Unsubscribe).
    reqData = (SubsReqDataT*) malloc( sizeof(SubsReqDataT) );

    // Попълват се полетата в структурата reqData.
    // Тези полета не трябва да се фиксирани в кода, а да се попълват от потребителя или да се извличат от файл.
    reqData->streamType = XETRA_PUBLIC_UNRELIABLE_MARKET_STREAM_TYPE;

    // Задава се версия на приложение, както е посочена във vld_val.h.
    reqData->applVersion        = XETR_AVN_071;
    reqData->subjectLength      = sizeof( XetraIsinSubjectT );
    reqData->subject            = &subscrSubject;

    // При абониране за Gap Notifications, стойностите трябва да са:
    // reqData->applVersion        = SUBJECT_GAPINFO_VERSION;
    // reqData->subjectLength      = strlen( SUBJECT_GAPINFO );
    // reqData->subject            = SUBJECT_GAPINFO;

    reqData->authorizationData      = (void*) NULL;
    reqData->authorizationDataLength    = 0;

    // В този момент трябва да се попълни структурата на приложната заявка.
    // В общия случай тези полета не са фиксирани в кода, а се попълват от потребител.
    memcpy(subscrSubject, "DE0001234567", ISIN_LEN);

    statusDataGlobal = (StatusDataT*) malloc( sizeof(StatusDataT) );

    // Повиква се VCI_Subscribe, за да се извърши реалното абониране за потока от данни.
    VCI_Subscribe  (    &reqControl,
                        reqData,
                        broadcastCallback,
                        NULL,
                        NULL,
                        statusDataGlobal
                   );
    // Оценява се дали повикването е било успешно или неуспешно
    ret_status_handling(statusDataGlobal);
}
```

4.10 Получаване на данни по абонамент

Следната извадка от кода илюстрира как в потребителското приложение може да бъде реализирана функция за обратно повикване, която да извършва асинхронна обработка на отговори на получените излъчвания на данни.

Примерен код на subscription response callback:

```
void broadcastCallback (    ReqCntrlT           *reqData,
                            CallBkAppDataT      *appData,
                            AppCntxtDataT       *appCntxtData,
                            StatusDataT     *statusDataGlobal
                            )
{
    int                     responseSize;
    XetraIsinSubjectT*      responseData;
    int                     subsID;
    // Структурата "brcSubject" на параметъра appData
    // съдържа темата на излъчваното съобщение,
    // както и subscription ID.
    // Тази тема е идентична с темата, използвана при абонирането за потока от данни,
    // с изключение на това, че метасимволите (wildcards) са заменени от реални стойности.
    // Темата на Gap Notification е  SUBJECT_GAPINFO.

    // Структурата appData съдържа указател към ответните данни и
    // размера на тези данни.
    responseSize = appData->appReqBlockSize;
    responseData = (XetraIsinSubjectT*) appData->appRespData;

    // В този момент могат да бъдат прочетени контекстуалните данни
    printf("appCntxtData.custData = %s\n", (char*)appCntxtData->custData);
    switch( statusDataGlobal->techComplCode )
    {
        // Тази функция не обработва всички кодове за изпълнение, които могат да се получат.
        case ELB_TECH_OK:
            // в този момент структурата на излъчването може да бъде прочетена и обработена
            // ...

            printf( "Успешно обработена\n" );
            break;

        case ELB_TECH_REQ_UNSUCCESSFUL:
            // При това изключение приложението е длъжно да реши, въз основа на детайлите за контекста
            // (декодиране на изключението, данни за заявка, търговска фаза и др.), дали заявката
            // трябва да бъде подадена повторно или не трябва.
            printf( "\n%s\n", statusDataGlobal->techComplText );
            // Кодът за функционално изпълнение предоставя детайлна информация:
            printf( "%s\n", statusDataGlobal->complText );
            break;
        case ELB_TECH_SUBSCRIPTION_DELETED:
            // Приложението може да направи опит за повторно подаване на заявката за абониране
        case ELB_TECH_INTERNAL_ERROR:
            // Приложението трябва да прекъсне връзката и да направи опит за рестартиране. Ако грешката
            // се повтори, тогава логовете за изключения трябва да се прегледат за повече информация по проблема.
        case ELB_TECH_PENDING_REQUEST_DELETED:
            // висящата заявка за абониране не е обработена успешно.
            break;
        case ELB_TECH_UNSUBSCRIBED:
            // Това показва успешно прекратяване на абонирането
        printf( "\n%s\n", statusDataGlobal->techComplText );
        break;
            case ELB_TECH_SUBSCRIBED:
            // това означава успешно абониране
```

```
            // subscription ID се записва в глобалната променлива
            subsID = ((SubsRespDataT*) appData->appRespData)->subsID;
            printf( "\n%s\n", statusDataGlobal->techComplText );
            break;
        default:
            printf( "\nUnknown Error! Connection Id: %d\n", reqData->connectionID );
            ако (appCntxtData->custBlockSize > 0)
            {
            printf("appCntxtData.custData = %s\n", (char*)appCntxtData->custData);
            }
            printf("Версия на приложението = %d\n", appData->applVersion);
        break;
    }
}
```

4.11 Прекратяване на абонамент за поток от данни

За да прекрати абонирането за потока от излъчвани данни, потребителското приложение трябва да използва входната точка VCI_Unsubscribe.

Примерен код на VCI_Unsubscribe:

```
void unsubscribe_function (void)
{
        // променливите се дефинират във VCI формат
        ReqCntrlT           reqControl;         // запис за контрол на заявката
        UnsubsReqDataT      reqData;            // блок от данни за заявка unsubscribe
        StatusDataT*        statusDataGlobal;   // статус на функционалното изпълнение от VCI

        // попълват се параметрите на reqControl
        strcpy(reqControl.appDescr, "Xetra");   // Това е незадължително описание.
        reqControl.dbApplID = xetraDbApplId;    // Ще бъде прекратен абонаментът за Xetra.
        strcpy(reqControl.VCIver, CVN_012);     // Посочва се версия на VALUES с цел проверка за
                                                //съвместимост;
                                                // VCI_VERSION е константа във VALUES.

        // В този момент connection ID, получен като върнат  параметър
        // от VCI_Connect, се използва за идентифициране на сесията.
        reqControl.connectionID = myConnectionID;

        // за прекратяване на абонирането е нужен глобално записаният subscription ID.
        reqData.subsID = subsId;

        statusDataGlobal = (StatusDataT*) malloc( sizeof(StatusDataT) );

        // Повиква се VCI_Unsubscribe
        VCI_Unsubscribe(    &reqControl,
                            &reqData,
                            NULL,
                            statusDataGlobal
                            );
        // Оценява се дали повикването е било успешно или неуспешно.
        ret_status_handling(statusDataGlobal);
}
```

4.12 Прекратяване на достъпа до Борсова услуга

За да прекрати достъпа до дадена Борсова услуга, потребителското приложение трябва да използва входната точка VCI_Logout.

Примерен код на VCI_Logout:

```
// Изваждане на първи потребител от Back End приложението.
// Структурата reqControl1 на типа ReqCntrlT съдържа контекстуални данни за влизането (login) на първия потребител
VCI_Logout(     &reqControl1,
                NULL,
                NULL,
                &statusData
                );

// Оценява се дали повикването е било успешно или неуспешно.
ret_status_handling(&statusData);

// В тази функция прослушваме VMQ и при събитие повикваме VCI_Dispatch.
poll_and_dispatch ( &reqControl1, fd );

// Изваждане на втори потребител.
// Структурата reqControl2 съдържа контекстуални данни за влизането (login) на втория потребител
VCI_Logout(     &reqControl2,
                NULL,
                NULL,
                &statusData
                );

// Оценява се дали повикването е било успешно или неуспешно.
ret_status_handling(&statusData);

// В тази функция прослушваме VMQ и при събитие повикваме VCI_Dispatch.
poll_and_dispatch ( &reqControl1, fd );
```

4.13 Прекратяване на сесия във VALUES

За прекратяване на връзката между потребителското приложение и VALUES трябва да се използва входната точка VCI_Disconnect.

Примерен код на VCI_Disconnect:

```
// заделя се памет за discnctreqData (тази памет трябва да се освободи
// след прекъсването на връзката)
discnctreqData = (DiscnctReqDataT*) malloc( sizeof(DiscnctReqDataT) );

// Повиква се VCI_Disconnect за прекратяване на сесията във VALUES.
VCI_Disconnect(     &reqControl1,
                    discnctreqData,
```

```
                        &statusData
                        );
// Оценява се дали повикването е било успешно или неуспешно.
ret_status_handling(statusData);

// Освобождава се паметта за данните, свързани с connection request
free(cnctreqData);
free(discnctreqData);
```

4.14 Спомагателни функции на потребителското приложение

Този раздел съдържа примери за някои спомагателни функции, тъй като потребителското приложение може да съдържа и такива функции. Те са представени тук, за да се придаде завършеност на примерната програма.

```
void error_handling ( StatusDataT* statusDataGlobal )
{
printf("Error: %s\n", statusDataGlobal->complText);
exit(statusDataGlobal->complCode);
}
void warning_handling ( StatusDataT* statusDataGlobal )
{
printf("Warning: %s\n", statusDataGlobal->complText);
}
void ret_status_handling ( StatusDataT* statusDataGlobal )
{
    // Оценява се дали повикването е било успешно или неуспешно.
    switch (statusDataGlobal->complSeverity)
    {
    case VCI_SUCCESS:     // VCI_SUCCESS = Успешно изпълнение.
        break;
    case VCI_FATAL:       // VCI_FATAL = Възникнала е фатална грешка.
                          // Приложението трябва да изпълни процедурата за затваряне (shutdown).
        fatality_handling(statusDataGlobal);
        break;
    case VCI_ERROR:       // VCI_ERROR = Открита е грешка.
                          // Приложението трябва да изпълнени процедура
                          // за обработка на грешка в зависимост от вида на
                          // получения код за изпълнение.
        error_handling(statusDataGlobal);
        break;
    case VCI_WARNING:     // VCI_WARNING = Възникнала е незначителна грешка.
                          // Приложението трябва да изпълнени процедура
                          // за обработка на грешка в зависимост от вида на
                          // получения код за изпълнение.
        warning_handling(statusDataGlobal);
        break;
    }  // end switch
    return;
}
```

5 VALUES API. Кодове за изпълнение (от интерфейса за повикване)

Интерфейсът за повикване на VALUES API съобщава информация за статуса посредством поле, съдържащо кодове за изпълнение. Всяка входна точка генерира множество различни кодове за изпълнение. Някои от кодовете са общи за входните точки.

Списъците на кодовете за функционално изпълнение, предоставяни от VALUES API, са дадени в томовете за съответните Борсови приложение. *Таблица 5.1* съдържа всички валидни кодове за техническо изпълнение, предоставяни от VALUES API, текстова дешифровка на кодовете и описание. За достъп до тези кодове и техните текстови дешифровки трябва да се използват дефинициите на константите в съответния заглавен файл (header file). Списък на заглавните файлове, предоставяни със софтуера на съответната Борсова услуга, е даден в *раздел 7*.

Текстова дешифровка на кода за изпълнение (techComplCode)	**Описание**
SUCCESSFUL COMPLETION УСПЕШНО ИЗПЪЛНЕНИЕ	Посочва статус на успешно изпълнение във входната точка.
MAXIMUM NUMBER OF CONNECTIONS REACHED ДОСТИГНАТ Е МАКСИМАЛНИЯТ БРОЙ СВЪРЗВАНИЯ	Достигнат е максимално допустимият брой сесии във VALUES за работната станция или интеграционния сървър (MISS).
USER IS NOT ALLOWED TO USE THE EXCHANGE SERVICE OR USER IS NOT REGISTERED ПОТРЕБИТЕЛЯТ НЯМА ПРАВО ДА ИЗПОЛЗВА БОРСОВАТА УСЛУГА ИЛИ ПОТРЕБИТЕЛЯТ НЕ Е РЕГИСТРИРАН	Посоченият user ID не е валиден.
INVALID PASSWORD НЕВАЛИДНА ПАРОЛА	Въведената парола не е валидна.
A PARAMETER WAS EITHER INVALID OR HAD INVALID CHARACTERS ПАРАМЕТЪРЪТ Е НЕВАЛИДЕН ИЛИ СЪДЪРЖА НЕВАЛИДНИ СИМВОЛИ	Един или повече от посочените параметри не са валидни. Молим вижте подробности в лог-файла за изключенията.
INVALID USER OR PASSWORD НЕВАЛИДЕН ПОТРЕБИТЕЛ ИЛИ ПАРОЛА	Един или повече от посочените параметри не са валидни. Молим вижте подробности в лог-файла за изключенията.
INVALID PARAMETER PASSED ПРЕДАДЕН Е НЕВАЛИДЕН ПАРАМЕТЪР	Един или повече от посочените параметри не са валидни. Молим вижте подробности в лог-файла за изключенията.

Текстова дешифровка на кода за изпълнение (techComplCode)	**Описание**
INTERNAL ERROR OCCURRED ВЪЗНИКНАЛА Е ВЪТРЕШНА ГРЕШКА	Възникнала е вътрешна грешка във VALUES. Молим вижте подробности в лог-файла за изключенията и ако проблемът не може да бъде решен, обърнете се към бюрото за помощ (help desk).
APPLICATION ALRREADY CONNECTED ПРИЛОЖЕНИЕТО ВЕЧЕ Е СВЪРЗАНО	Вече има установена сесия.
REQUEST NOT SUCCESSFULLY PROCESSED НЕУСПЕШНО ИЗПЪЛНЕНИЕ НА ЗАЯВКАТА	Възникнало е изключение при обработката на заявката. За подробности молим вижте кодовете за изпълнение на съответното Борсово приложение.
APPLICATION NOT CONNECTED ПРИЛОЖЕНИЕТО НЕ Е СВЪРЗАНО	На първо място трябва да се установи сесия.
PENDING REQUEST DELETED ИЗТРИТА Е ВИСЯЩА ЗАЯВКА	Приложението е прекъснало връзката или е излязло от Борсова услуга, докато все още е имало висящи заявки. Алтернативно, може да е задействано непрозрачно прекъсване, заявките трябва да бъдат предадени повторно.
SUBSCRIPTION DELETED АБОНАМЕНТЪТ Е ИЗТРИТ	Приложението е прекъснало връзката, докато все още е имало активни абонирания. При Xervices с Broadcast Extension тази грешка може да възникне и при загуба на login-контекста, за който се отнася абонаментът.
MAXIUM NUMBER OF LOGINS EXCEEDED ПРЕВИШЕН Е РАЗРЕШЕНИЯТ МАКСИМАЛЕН БРОЙ ВЛИЗАНИЯ	Превишен е максималният брой влизания (logins), разрешен за една Борсова услуга.
EXCHANGE SERVICE NOT AVAILABLE НЕРАЗПОЛАГАЕМОСТ НА БОРСОВАТА УСЛУГА	Борсовата не е налична в момента. Молим свържете се с оператора.
TOO MANY PENDINGREQUESTS IN QUEUE В ОПАШКАТА ИМА ПРЕКАЛЕНО МНОГО ВИСЯЩИ ЗАЯВКИ	Достигнат е максимално разрешеният брой висящи заявки. Изчакайте отговорите, преди да въвеждате нови заявки.
USER NOT LOGGED IN ПОТРЕБИТЕЛЯТ НЕ Е ПОЛУЧИЛ ДОСТЪП	Потребителят още не е получил оторизация.
TECHNICAL SERVICES NOT AVAILABLE НЕРАЗПОЛАГАЕМОСТ НА ТЕХНИЧЕСКИТЕ УСЛУГИ	Неразполагаемост на GATE (технически услуги). Молим свържете с оператора.

Текстова дешифровка на кода за изпълнение (techComplCode)	**Описание**
USER LOGGED IN SUCCESSFULLY ПОТРЕБИТЕЛЯТ Е ПОЛУЧИЛ ДОСТЪП УСПЕШНО	Потребителят е оторизиран успешно.
USER LOGGED OUT SUCCESSFULLY ДОСТЪПЪТ НА ПОТРЕБИТЕЛЯ Е ПРЕКРАТЕН УСПЕШНО	Потребителят успешно е излязъл от Борсовата услуга.
STREAM SUBSCRIBED ОСЪЩЕСТВЕНО Е АБОНИРАНЕ ЗА ПОТОКА	Посоченият абонамент е активиран.
STREAM UNSUBSCRIBED АБОНИРАНЕТО ЗА ПОТОКА Е ПРЕКРАТЕНО	Посоченият абонамент е деактивиран.
UNMAPPABLE MESSAGE DISCRADED ОТСТРАНЕНО Е НЕРАЗПОЗНАВАЕМО СЪОБЩЕНИЕ	Получен е отговор, за който няма съответстваща висяща заявка. Това събитие може да се случи при възстановяването на работна станция или MISS (прекъсване).
NONTRANSPARENT FAILOVER НЕПРОЗРАЧНО ПРЕКЪСВАНЕ	Възникнало е непрозрачно прекъсване към вторичния MISS. Може някои заявки да са пропаднали и да се наложи възстановяване чрез издирване ((inquiry).
VALUES CALLS ARE NOT REENTRANT ПОВИКВАНИЯТА КЪМ VALUES НЕ СА ПОВТОРЯЕМИ	Направен е опит за използване на входна точка на VALUES, докато е активно друго повикване към VALUES в същия процес.
EXCHANGE SERVICE AVAILABLE БОРСОВАТА УСЛУГА Е РАЗПОЛАГАЕМА	Борсовата услуга отново е разполагаема
STREAM NOT SUBSCRIBED НЯМА АБОНАМЕНТ ЗА ПОТОКА	Абонаментът не е активен.

Таблица 5.1: Кодове за изпълнение, връщани от VALUES API.

6 VALUES API. Интерфейс за повикване. Описание на полетата

В този раздел са описани полетата с данни, които се предават до интерфейса за повикване на VALUES API или се извличат от този интерфейс. В първата точка е представена обзорна и обща информация. Във втората точка са описани всички полета с данни на интерфейса за повикване.

6.1 Обзор

Този раздел съдържа списък с наименованията и подробни описания на всички полета с данни във VALUES API. Обяснени са характеристиките на полетата и са дадени указания за инициализирането им. Където е необходимо са дадени валидните стойности за отделните полета и текстови обяснения на правилата, приложими на нивото на полета.

6.1.1. Характеристики на полетата

Характеристиките на всяко поле са представени чрез следната информация:

Тип — Посочва типа на данните в полето. Валидни са ANSI C типовете, дадени в *таблица 6.1.*

Стойност/референция — Определя механизма за предаване, който се използва за съответното поле – чрез стойност (Value) или чрез референция (Reference).

Тип	Описание	Валидни стойности или валиден диапазон
int	Цяло число, обикновено отразява естествената величина на целите числа в приемната машина	Зависят от платформата (p/d)
char [ ]	Текст, в C това е референция към текста, съхраняван в даден масив (array). 'Format' указва размера на масива.	Букви и цифри, не са разрешени специални символи
char	Един байт, който може да поеме един символ в локалния набор от символи.	Зависят от платформата (p/d)
void*	Референция към блок от данни с неопределена структура	Зависят от платформата (p/d)

Таблица 6.1: Типове данни във VALUES API

Формат — Определя форма́та и размера на полето с данни. Форматът за типовете, чийто размер зависи от платформата, е обозначен с "p/d".

Форматните константи са дадени в заглавните файлове (header files) за дефиниране на данните, които са публикувани със

софтуерния продукт VALUES API. Списък на заглавните файлове за дефиниране на данните във VALUES API е даден в *раздел 7*.

Бележка: Когато стойностите, които се използват в приложните заявки, се изразяват чрез символи, тези стойности не завършват с '\0'.

6.1.2. Указания за инициализиране

Препоръчва се всички неизползвани или незадължителни полета с данни да се инициализират, преди да бъдат изпратени във VALUES API. Указания за инициализирането на различните полета с данни са дадени в *таблица 6.2*. За неизползваните полета със символи в приложните заявки задължително се задават интервали ("spaces").

Тип	Инициализация
int	0
char [ ]	Масив, попълнен с интервали ("spaces")
char	SPACE
void*	NULL

Таблица 6.2: Указания за инициализиране на полетата с данни

6.1.3. Шаблон за описване на полетата от интерфейса за повикване

Наименование на полето			
Описание	Описание на полето от интерфейса за повикване		
Къде се използва	Входните точки, за които се използва съответното поле		
Характеристики	Тип	Стойност/Референция	Формат
Валидни стойности			

6.2 Описание на полетата в интерфейса за повикване

6.2.1. appDescr (ReqCntrlT)

Описание	Това поле е резервирано за по-късна употреба		
Къде се използва	n/a		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	MAX_APPDSCR
Валидни стойности	Не се отнася за това поле.		

6.2.2. applClass (XerviceInfoT)

Описание	Това поле посочва класа, определен за дадена Xervice (Xervice Class). Концептуално описание на класовете Xervice е дадено в *раздел 2.9*.		
Къде се използва	При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Дефинирани са в exid.h, виж *раздел 7* за информация относно дефинициите на данните и заглавните файлове.		

6.2.3. applPrevVersion (CallBkAppDataT)

Описание	Това поле посочва AVN, за който Борсовото приложение поддържа обратна съвместимост. Ако не се поддържа обратна съвместимост, стойността на това поле е еднаква със стойността на полето applVersion.		
Къде се използва	При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Виж описанията на това поле в томовете за отделните Борсови приложения.		

6.2.4 applVersion (LoginReqDataT, SubsReqDataT, CallBkAppDataT, XerviceInfoT)

Описание	Това поле идентифицира версията на приложната заявка (AVN).		
Къде се използва	VCI_Subscribe VCI_Login При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Виж описанията на това поле в томовете за отделните Борсови приложения.		

6.2.5 appReq (SubmitReqDataT)

Описание	Това е указател (pointer) към структурата с данни за приложната заявка, предавана на VCI_Submit.		
Къде се използва	VCI_Submit		
Характеристики	Тип	Стойност/Референция	Формат
	void*	Reference	p/d
Валидни стойности	Данни за приложната заявка в зависимост от конкретното Борсово приложение.		

6.2.6 appReqBlockSize (CallBkAppDataT, SubmitReqDataT)

Описание	Това поле посочва броя байтове на блоковете с данни за приложната заявка, предаван на VCI_Submit от потребителското приложение. Това поле се връща от VALUES в application callback.
Къде се използва	VCI_Submit При обратни повиквания към приложението.

Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d

Валидни стойности: Реалният размер на структурата с данни за приложната заявка.

6.2.7 appReqData (CallBkAppDataT)

Описание: Тази структура с данни се връща от VALUES в application callback при получаване на отговори. Съдържа приложната заявка, за която се отнася отговорът.

Къде се използва: При обратни повиквания към приложението.

Характеристики	Тип	Стойност/Референция	Формат
	void*	Reference	p/d

Валидни стойности: Данни за приложната заявка в зависимост от конкретното Борсово приложение.

6.2.8 appRespBlockSize (CallBkAppDataT)

Описание: Това поле показва броя байтове на блока с данни, съдържащ отговора на приложна заявка (приложния отговор), който се попълва от VALUES и се предава на потребителското приложение чрез application callback.

Къде се използва: При обратни повиквания към приложението.

Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d

Валидни стойности: Реалният размер на структурата с данни, съдържаща отговора на приложната заявка.

6.2.9 appRespData (CallBkAppDataT)

Описание	Това е указател към структурата с данни, съдържаща отговор на приложна заявка, която се попълва от VALUES и се предава на потребителското приложение чрез application callback.		
Къде се използва	При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	void*	Reference	p/d
Валидни стойности	Приложни данни в зависимост от конкретното Борсово приложение.		

6.2.10 authorizationData (LoginReqDataT, SubsReqDataT)

Описание	Това е указател към оторизационните данни за съответното Борсово приложение. Използва се за пароли и друга допълнителна информация, специфична за Борсовото приложение.		
Къде се използва	VCI_Login VCI_Subscribe (за по-късна употреба)		
Характеристики	Тип	Стойност/Референция	Формат
	void*	Reference	p/d
Валидни стойности	Данни за оторизация в зависимост от конкретното Борсово приложение.		

6.2.11 authorizationDataLength (LoginReqDataT, SubsReqDataT)

Описание	Това поле посочва дължината на полето authorizationData.		
Къде се използва	VCI_Login		
	VCI_Subscribe (за по-късна употреба)		
Характеристики	Тип	Стойност/Референция	Формат
	void*	Reference	p/d
Валидни стойности	Размер на полето authorizationData.		

6.2.12 brcSubject (CallBkAppDataT)

Описание	Това поле посочва структурата с данни за абонаментната тема. Тази структура се попълва от потребителското приложение и се предава на VALUES като байт-блок. Структурата с данни за абонаментната тема се дефинира с всяка заявка за абониране, както е описано в тома за съответното Борсово приложение.		
Къде се използва	VCI_Subscribe		
	При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	void*	Reference	p/d
Валидни стойности	Дефиниции на структурите са данни за абонаментните теми са дадени в томовете на съответните Борсови приложения. Тези структури се дефинират в описанието на всяка заявка за абониране.		

6.2.13 closure (LoginReqDataT)

Описание	Това поле е резервирано за по-късна употреба		
Къде се използва	n/a		
Характеристики	Тип	Стойност/Референция	Формат
	char	Value	1
Валидни стойности	Не се отнася за това поле.		

6.2.14 complCode (statusDataT)

Описание	Това поле съдържа код, който описва статуса на изпълнение на приложната заявка. Статусът за изпълнение се генерира от Борсовата услуга и се предава на application callback чрез VALUES.		
Къде се използва	Във всички входни точки на интерфейса за повикване.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Стойността варира според подадената приложна заявка. Списъци с кодовете за изпълнение на подадените приложни заявки и текстови дешифровки на тези кодове са дадени в томовете за отделните Борсови приложения.		

6.2.15 complSeverity (statusDataT)

Описание	Това поле обозначава успешното изпълнение или сериозността на евентуалната грешка за съответния complCode. Стойността му варира според обстоятелствата на извършената обработка.		
Къде се използва	Във всички входни точки на интерфейса за повикване.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	VCI_SUCCESS		
	VCI_WARNING		
	VCI_ERROR		
	VCI_FATAL		

6.2.16 complText (statusDataT)

Описание	Това поле дешифрира съответния код за изпълнение. Представлява текстово описание на кода за изпълнение.		
Къде се използва	Във всички входни точки на интерфейса за повикване.		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	ELB_MAX_STRING
Валидни стойности	Стойността варира според подадената приложна заявка. Списъци с кодовете за изпълнение на подадените приложни заявки и текстови дешифровки на тези кодове са дадени в томовете за отделните Борсови приложения.		

6.2.17 connectionID (ReqCntrlT, CnctRespDataT)

Описание	Това поле идентифицира сесията. Тя е уникална за всяко приложение, което използва VALUES. Полето се попълва от VALUES в съобщението-отговор на входната точка VCI_Connect. За всички останали входни точки полето се попълва от приложението в блока от данни за контрол на заявката, използвайки connection ID, получен чрез VCI_Connect.		
Къде се използва	Във всички входни точки на интерфейса за повикване.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Валиден connection ID, получен чрез VCI_Connect.		

6.2.18 custBlockSize (AppCntxtDataT)

Описание	Това поле посочва броя байтове на блока с индивидуални данни (поле custData), който се предава на VALUES от потребителското приложение и се връща чрез application callback.		
Къде се използва	VCI_Connect VCI_Login VCI_Submit VCI_Subscribe При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Цифрови стойности.		

6.2.19 custData (AppCntxtDataT)

Описание	Това поле съдържа указател към блока с индивидуални данни, който се предава на VALUES от потребителското приложение и се връща чрез application callback.		
Къде се използва	VCI_Connect VCI_Login VCI_Submit VCI_Subscribe При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	void*	Reference	p/d
Валидни стойности	Специфични за потребителя контекстуални данни.		

6.2.20 dbApplID (ReqCntrlT)

Описание	Това поле идентифицира по уникален начин Борсовата услуга, която е предмет на операциите по влизане, излизане, подаване на заявки или абониране. Полето се попълва от функцията connect callback и потребителското приложение.		
Къде се използва	VCI_Login VCI_Logout VCI_Submit VCI_Subscribe При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Всяко цяло число, получена от функцията connect callback (за подробности виж *точка 3.3.3*).		

6.2.21 exchApplId (XerviceInfoT)

Описание	Това поле съдържа Market Identification Code (MIC), който идентифицира по уникален начина дадена Борса, например "XEUR" за Eurex или "XVIE" за Xetra Vienna.		
Къде се използва	При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	MAX_APPLID
Валидни стойности	Валидни стойност на MIC. Молим да се има предвид, че това поле не завършва с '\0'.		

6.2.22 exchDscrName (XerviceInfoT)

Описание	Това поле съдържа четим идентификатор на съответната Борса във формата на свободен текст. Може да се използва за показване на меню пред крайния потребител.		
Къде се използва	При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	MAX_DSCRNAME
Валидни стойности	Свободен текст. Полето съдържа С-низ, завършващ с '\0'.		

6.2.23 funcResult (LoginRespDataT)

Описание	Това поле е резервирано за по-късна употреба		
Къде се използва	n/a		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	n/a
Валидни стойности	Не се отнася за това поле.		

6.2.24 loginID (ReqCntrlT, LoginRespDataT)

Описание	Това поле съдържа уникален идентификатор (loginID) на всеки успешно създаден контекст за достъп до Борсова услуга (влизане). Този loginID е уникален за всяка Борсова услуга. Полето се попълва от VALUES и се връща на потребителското приложение с отговора на успешна заявка за влизане. Така полученият loginID се предава от потребителското приложение на VALUES с всяко повикване на VCI_Submit или VCI_Logout. Това се отнася и за VCI_Subscribe, ако съответната Xervice реализира Broadcast Extension. Полето се използва за установяване на автентичността на конкретния потребител.		
Къде се използва	VCI_Login VCI_Logout VCI_Submit		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Валиден loginID, получен от VCI_Login.		

6.2.25 password (CnctReqDataT)

Описание	Това поле се използва за оторизиране (разрешаване) на връзката с VALUES. Паролата се посочва от потребителско приложение и се предава на VALUES. В това поле могат да се въвеждат всички символи, които са разрешени за системни пароли на платформата, върху която работи MISS.		
Къде се използва	VCI_Connect		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ] (виж по-горе за специалните символи	Value	MAX_PWDID
Валидни стойности	Валидна парола, която е регистрирана на MISS.		

6.2.26 prodMode (CnctRespDataT)

Описание	Това поле посочва експлоатационния режим на Борсовите приложения: реална (производствена експлоатация), симулация или развой. Полето се връща на потребителското приложение чрез отговора на VCI_Connect.		
Къде се използва	VCI_Connect		
Характеристики	Тип	Стойност/Референция	Формат
	char	Value	1
Валидни стойности	VCI_AREA_PROD		
	VCI_AREA_SIM		
	VCI_AREA_DEV		

6.2.27 reqID (ReqCntrlT)

Описание	Това поле посочва приложната заявка, подавана чрез VCI_Submit в контекста на дадена Борсова услуга. На всяка приложна заявка се присвоява уникален request ID. Потребителското приложение посочва съответния request ID.		
Къде се използва	VCI_Submit		
	При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Списъци с всички възможни стойности на request ID са дадени в томовете за отделните Борсови приложения.		

6.2.28 resubmitFlag (ReqCntrlT)

Описание	Този флаг посочва дали повторно подадена приложна заявка е била обработена. Ако заявката е била обработена, полето получава стойност RESUBMISSION_PROCESSED и данните за обработката са налични в отговора към приложението. Ако заявката не е била обработена – поради грешка или поради това, че обработката вече е извършена в отговор на друга инстанция на заявката, тогава полето получава стойност RESUBMISSION_NOT_PROCESSED и отговорът към приложението не съдържа данни.		
Къде се използва	При обратни повиквания към приложението. Виж тома за съответното Борсово приложение.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	RESUBMISSION_PROCESSED		
	RESUBMISSION_NOT_PROCESSED		

6.2.29 resubmitNo (ReqCntrlT)

Описание	Това поле идентифицира по уникален начин повторно подадена приложна заявка.		
Къде се използва	VCI_Submit.		
	При обратни повиквания към приложението. Виж тома за съответното Борсово приложение.		
Характеристики	Тип	Стойност/Референция	Формат
	unsigned int	Value	p/d
Валидни стойности	Цифрови		

6.2.30 streamType (CallbkAppDataT, SubsReqDataT)

Описание	Това поле посочва потока от излъчвани данни, за който приложението иска да се абонира. Всеки поток от данни дефинира собствен streamType, който е уникален в контекста на съответното Борсово приложение. Това поле се попълва от потребителското приложение и се предава на VALUES.		
Къде се използва	VCI_Subscribe При обратни повиквания към приложението.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Валидните типове потоци са дадени във файловете за дефиниране на данните, публикувани със софтуерния продукт VALUES API. Виж *раздел 7* за информация относно файловете за дефиниране на данните. Освен това, виж тома на съответното Борсово приложение за допълнителна информация относно абонираният и типовете потоци от излъчвани данни.		

6.2.31 subject (CallBkAppDataT)

Описание	Това поле е резервирано за по-късна употреба		
Къде се използва	n/a		
Характеристики	Тип	Стойност/Референция	Формат
	BcastSubjectT*	n/a	n/a
Валидни стойности	Не се отнася за това поле.		

6.2.32 subject (SubsRegDataT)

Описание	Това поле дефинира структурата с данни за абонаментната тема. Тази структура се попълва от потребителското приложение и се предава на VALUES като байт-блок. Структурата с данни за абонаментната тема се дефинира с всяка заявка за абониране, както е описано в тома за съответното Борсово приложение.		
Къде се използва	VCI_Subscribe		
Характеристики	Тип	Стойност/Референция	Формат
	void*	Reference	p/d
Валидни стойности	Дефиниции на структурите с данни за абонаментните теми са дадени в томовете на съответните Борсови приложения. Тези структури се дефинират в описанието на всяка заявка за абониране.		

6.2.33 subjectLength (SubsReqDataT)

Описание	Това поле дефинира дължината на абонаментната тема в байтове. Полето се попълва от потребителското приложение и се предава на VALUES.		
Къде се използва	VCI_Subscribe		
Характеристики	Тип	Стойност/Референция	Формат
	int	Reference	p/d
Валидни стойности	Реалният размер на структурата с данни за абонаментната тема.		

6.2.34 subsID (CallBkAppDataT, SubsRespDataT)

Описание	Това поле идентифицира по уникален начин съответния абонамент. Полето се попълва от VALUES в отговора на успешна заявка за абониране. Когато иска да прекрати абонирането за даден поток от данни, потребителското приложение трябва да посочи съответния subsID.		
Къде се използва	VCI_Unsubscribe При обратни повиквания към приложението		
Характеристики	Тип	Стойност/Референция	Формат
	int	Reference	p/d
Валидни стойности	Валиден subsID, получен от VCI_Subscribe.		

6.2.35 subsSubject (SubsRegDataT)

Описание	Това поле е резервирано за по-късна употреба		
Къде се използва	n/a		
Характеристики	Тип	Стойност/Референция	Формат
	BcastSubjectT*	n/a	n/a
Валидни стойности	Не се отнася за това поле.		

6.2.36 techComplCode (StatusDataT)

Описание	Това поле съдържа код, който описва статуса на изпълнение в интерфейса за повикване или GATE. Полето се попълва от VALUES при връщане на повикване към интерфейса.		
Къде се използва	Във всички входни точки на интерфейса за повикване.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	Стойността варира според входната точка на интерфейса за повикване. Списък на кодовете за изпълнение е даден в *раздел 5*.		

6.2.37 techComplSeverity (StatusDataT)

Описание	Това поле обозначава успешното изпълнение или сериозността на евентуалната грешка за съответния complCode. Стойността му варира според обстоятелствата на извършената обработка, например успешната обработка генерира VCI_SUCCESS.		
Къде се използва	Във всички входни точки на интерфейса за повикване.		
Характеристики	Тип	Стойност/Референция	Формат
	int	Value	p/d
Валидни стойности	VCI_SUCCESS		
	VCI_WARNING		
	VCI_ERROR		
	VCI_FATAL		

6.2.38 techComplText (StatusDataT)

Описание	Това поле дешифрира съответния код за изпълнение. Представлява текстово описание на кода за изпълнение.		
Къде се използва	Във всички входни точки на интерфейса за повикване.		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	ELB_MAX_STRING
Валидни стойности	Стойността варира според входната точка от интерфейса за повикване. Списъци с кодовете за изпълнение и текстовите дешифровки на тези кодове са дадени в *раздел 5*.		

6.2.39 userID (CnctReqDataT)

Описание	Това поле съдържа данни за идентифициране на потребителя при разрешаването на достъп до VALUES. Полето се попълва от потребителското приложение и се предава на VALUES.		
Къде се използва	VCI_Connect		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	MAX_USRID
Валидни стойности	Валидно име на потребител (user ID), което е известно на MISS.		

6.2.40 userID (LoginReqDataT)

Описание	Това поле съдържа данни за идентифициране на потребителя при разрешаването на достъп до Борсова услуга. Полето се попълва от потребителското приложение и се предава на VALUES.		
Къде се използва	VCI_Login		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	LOGIN_MAX_USRID
Валидни стойности	Валидно име на потребител (user ID), което е известно на Борсовата услуга.		

6.2.41 userID (SubsReqDataT)

Описание	Това поле съдържа данни за идентифициране на потребителя при разрешаването на абонамент за потоци от данни, излъчвани от съответното Борсово приложение. Полето се попълва от потребителското приложение и се предава на VALUES.		
Къде се използва	VCI_Subscribe		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	SUBS_MAX_USRID
Валидни стойности	Валидно име на потребител (user ID), което е известно на Борсовото приложение (за по-късна употреба).		

6.2.42 VCIver (ReqCntrlT)

Описание	Това поле съдържа номер на версията (CVN) на интерфейса за повикване на VALUES API. CVN се използва за поддържане на обратната съвместимост на интерфейса за повикване.		
Къде се използва	Всички входни точки на интерфейса да повикване.		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	MAX_VCVER
Валидни стойности	CVN_012		

6.2.43 VMQname (CnctRespDataT)

Описание	Това поле идентифицира опашката за съобщения във VALUES. Полето се попълва от VALUES чрез входната точка VCI_Connect. Потребителското приложение използва VMQname за прослушване на опашката за съобщения във VALUES, т.е. за да проверява дали има висящи събития.		
Къде се използва	VCI_Connect		
Характеристики	Тип	Стойност/Референция	Формат
	char [ ]	Value	MAX_VCVER
Валидни стойности	Подравнено отляво и завършващо с '\0' име на опашка.		

7. Дефиниции на данните

В този раздел е представен списък на заглавните файлове (header files) във VALUES, които са необходими за използване на интерфейса за повикване на VALUES API (технически компоненти). Заглавните файлове съдържат всички константи, номерационни типове и дефиниции на структурите, необходими за използване на интерфейса за повикване на VALUES API. Отделните заглавни файлове са описани в *таблица 7.1*. Всички заглавни файлове се предоставят в електронен формат със софтуерния пакет за съответното Борсово приложение. Заглавните файлове, необходим за изпращане на приложни заявки към VALUES API, са описани в томовете на отделните Борсови приложения.

Файл	Описание
Values.h	Съдържа прототипи на функциите от интерфейса за повикване, дефиниции на константите и дефиниции на други структури, необходими за използване на интерфейса за повикване на VALUES API.
elbcodetech.h	В този заглавен файл са събрани кодовете за всички технически изключения, които може да върне VALUES API. Дефинирани са константите за всеки от тези кодове. Кодовете на техническите изключения са дешифрирани във файла msg.dat, който е включен в разпространявания софтуерен пакет за VALUES API.
subject.h	Съдържа дефиниции на константите при абониране (като метасимволи за абонаментни теми, специалната тема GAPINFO).
exid.h	Съдържа дефиниции на предпроцесорните константи за Xervice Classes. Предпроцесорните константи за отделни Xervices, които са остарели след въвеждането на концепцията за Xervice Classes, ще бъдат отстранени от файла при излизането на нова версия. Тези константи не трябва да се използват повече.

Таблица 7.1: Заглавни файлове за VALUES API

8. Речник

Термин	Описание
API	Application Programming Interface Приложен програмен интерфейс
Application Callback Функция за обратно повикване	Функции на потребителското приложение, които се активират от VALUES при получаване на отговори на подадени от приложението заявки или на други събития. Потребителското приложение определя коя функция за обратно повикване да бъде активирана, като "регистрира" тази функция.
Application Dispatch Loop Диспечерски цикъл на приложението	Диспечерският цикъл е реализиран в потребителското приложение (обикновено това е основният обработващ цикъл). Този цикъл получава известия (нотификации) за събития във VALUES от VMQ, които се записват там от Борсовите приложения. Примери за видовете събития, които Борсовите приложения записват във VMQ, са отговорите, абонираният, известията и изключенията. След като получи известие за събитие във VMQ, диспечерският цикъл на приложението задейства VCI_Dispatch, за да извлече съответното събитие от VMQ.
Application request Приложна заявка	Приложните заявки са функционални входни точки към услугите, предоставяни от Борсовите приложения. Приложните заявки винаги се използват заедно с интерфейса за повикване на VALUES API, който осигурява функционалната информация, необходима за достъп до услугите на Борсовите приложения.
Application response Приложен отговор (Отговор на приложна заявка)	Приложните отговори се получават от потребителското приложение като индивидуални и единични реакции на конкретни заявки. Приложният отговор съдържа резултатите от обработката на заявката в Борсовото приложение. Приложният отговор се предава асинхронно на потребителското приложение.
Asynchronous processing Асинхронна обработка	Асинхронната входна точка не блокира, докато трае обработката. Асинхронните входни точки незабавно връщат на потребителското приложение статус на обработката на заявката. Отговорите с резултатите от обработките се получават в по-късен момент (т.е. асинхронно спрямо заявката за обработка).
AVN	Application Version Number (номер на версията на дадено приложение). Всички приложни заявки, заявки за абониране, приложни отговори и излъчвания на данни съдържат AVN, който дефинира версията на съответното Борсово приложение.
Back End	Общ термин за всяка система на Deutsche Börse, която предлага централизирани услуги на участниците, имащи достъп до нея (например Xetra®).

Термин	Описание
BESS	Back End related Subsystem. Подсистема на MISS за конкретна борса.
Broadcast Излъчване (на данни)	Излъчването представлява информация, изпращана от дадена борса до участниците в търговията на тази борса (напр. потвърждение на сделка, потвърждение на поръчка). Излъчваните данни се изпращат чрез различни потоци, като всеки поток от данни съдържа специфична информация.
Broadcast gap stream Поток за пропуснати излъчвания	При излъчванията на различни видове потоци от данни могат да се случат прекъсвания. Потокът за пропуснати излъчвания може да се използва за откриване на липсващи излъчвания и за уведомяване на приложенията, че е възможно някои излъчвания да са били загубени. Използването на темата Gap Info (за подробности виж *точка 2.7*) е препоръчително винаги, когато има възможност.
Call Interface Интерфейс за повикване	Интерфейсът за повикване съдържа техническите входни точки към услугите, предоставяни от Борсовите приложения. Интерфейсът за повикване е набор от системни операции, които се повикват от приложението за реализиране на достъп до тези услуги.
Completion code Код за изпълнение	Интерфейсът за повикване на VALUES API съобщава информация за статуса посредством поле, съдържащо код за изпълнение. Всяка входна точка може да генерира няколко различни кода за изпълнение.
CCP ЦД	Central Counterparty Централен депозитар
CVN	Call Interface Version Number. Всяка входна точка от интерфейса за повикване съдържа CVN.
Data streams Потоци от данни	Потоците от данни са средството за разпространяване на информация от определен вид до участниците на борсовия пазар. Всеки тип информация дефинира собствен поток от данни.
DBAG	Deutsche Börse AG
End user application Приложение на краен потребител, потребителско приложение	Това е всяко приложение, което получава услуги чрез VALUES API.
Entry point Входна точка	Интерфейсът за повикване на VALUES API се състои от фиксиран брой технически входни точки, които се използват за започване на сесия, изпращане на приложни заявки, заявяване на данни и получаване на отговори.
Eurex	European Exchange. Система за електронна търговия и клиринг на пазара за производни инструменти (опции и фючърси).
Eurex US	Поделението на Eurex в САЩ.

Термин	Описание
Exception event Събитие-изключение	Събитие, изпращано до VALUES API при възникване на извънредно обстоятелство по време на връзката (например неизправности по мрежата, неизправност на Борсово приложение и др.).
Exchange Service Борсова услуга	Виж Борсово приложение.
Field usage Правило за използване на поле	Правилото за използване на поле определя дали дадено поле е задължително, незадължително или се инстанцира многократно. Правилата за използване на полетата се посочват в описанията на приложните заявки.
Front End Интерфейсна (клиентска) система	Всяка една локална система на участник на борсовия пазар, която осигурява достъп до Борсовите приложения.
GATE	GATE (Generic Access To Exchanges – Единен достъп до борсите) е съвкупност от процеси, които се изпълняват върху работната станция и MISS на участника. Тези процеси осигуряват техническата структура за VALUES и комуникацията с Борсовите приложения.
MAA	Member Assembled Application (приложение на участник). Потребителско приложение, създадено от външен разработчик, което работи върху VALUES.
Member Участник	Участник на борсовия пазар.
Member application Приложение на участник	Виж MAA.
MISS	Member Integration System Server Интеграционен сървър на участник
Paging разбивка на данните и командите по блокове (страници)	Механизмът за paging във VALUES API позволява обработката на отговори, съдържащи големи обеми от данни, като едновременно с това се редуцира размерът на предаваните съобщения.
Request ID	Идентификатор, който индивидуализира всяка една приложна заявка до Борсова услуга.
Request Management Service Услуга за управление на заявките	Услуга, която осигурява достъп до Борсова услуга. Всяка една отработена заявка се състои една приложна заявка и един приложен отговор. Използва се за задействане на обработки в Борсовата услуга.
Response Event Събитие-отговор	Отговор на заявка, изпратена от потребителско приложение до Борсова услуга.

Термин	Описание
Session Сесия	Сесията във VALUES е техника за управление на комуникациите между потребителското приложение и VALUES с използване на GATE. Комуникациите между потребителските и Борсовите приложения са базирани на сесии. За един приложен процес може да се открива само една сесия във VALUES.
Subscription Абониране, абонамент	Абонирането е механизъм за поискване на управлявана от събития информация, предоставяна от Борсовите приложения. Абонирането може да бъде стартирано и спирано. Поисканата абонаментна информация пристига асинхронно. Заявките за абониране се изпращат до Борсовите приложения, като задължително определят желания поток от данни. След абонирането всички нови излъчвания на данни от посочения поток се изпращат асинхронно на потребителското приложение.
Synchronous processing Синхронна обработка	Синхронните входни точки от интерфейса за повикване блокират, докато завърши вътрешната обработка във VALUES, т.е. отработването на подадена заявка приключва с получаването на отговор с резултатите от обработката.
Technical Service Технически услуги	Виж GATE.
User Потребител	За спесификацията на този интерфейс терминът "потребител" означава всеки един член, търговец или участник, който ползва Борсови услуги чрез VALUES API.
User Interface Event Queue Опашка от събития на потребителския интерфейс	Опашка от събития, които управляват функционирането на потребителското приложение. Например графичният потребителски интерфейс (GUI) използва опашка от събития, за да управлява входните сигнали, получавани от клавиатурата или мишката.
VALUES	Virtual Access Link Using Exchange Services Application Programming Interface. Връзка за виртуален достъп до Борсови услуги с използване на API.
VALUES API	VALUES API представлява набор от системни операции и структури от данни за осъществяване на комуникация с Борсовите услуги.
VALUES events Събития във VALUES	Събитието във VALUES е индикация за получаването на приложни отговори, излъчвания, известия или изключения от опашката за съобщения във VALUES (VMQ).
VCI	VALUES Call Interface. Виж "Интерфейс за повикване"
VMQ	Values Message Queue. Опашка за временно съхранение на данни за комуникационния канал, приложните заявки, приложните отговори и излъчванията, които се предават между VALUES и GATE.

Термин	Описание
WS PC	Workstation Работна станция
Xervice	Услуга, предоставяна от дадена електронна борса на интерфейсна (клиентска) система, която е базирана на VALUES. Дадена борса може да предлага няколко Xervices. Виж *точка 2.9* за повече подробности.
Xervice Class Клас Xervice	Класът Xervice е идентификатор, който е единен за всички Xervices, предлагащи идентична функционалност на VALUES, при условие, че номерът на версията на приложението (Application Version Number) също е еднакъв. Виж *точка 2.9* за повече подробности.
Xetra®	Exchange Electronic Trading. Електронна система на Deutsche Börse за търговия на спот-пазарите.
Xontro	Система за присъствена търговия, поддържана от фирмата-оператор BrainTrade.

Таблица 8.1: Речник