SERVICE IMPLEMENTATION METHOD, APPARATUS, AND SYSTEM BASED ON FIX PROTOCOL

[0001] The present application claims priority to Chinese Patent Application No. 201610112705.1, filed on February 29, 2016 and entitled "SERVICE IMPLEMENTATION METHOD, APPARATUS, AND SYSTEM BASED ON FIX PROTOCOL", which is incorporated here by reference in its entirety.

TECHNICAL FIELD

[0002] The present disclosure relates to the field of Internet technologies, and in particular, to a service implementation method, apparatus, and system based on the FIX protocol.

BACKGROUND

[0003] The Financial Information Exchange (FIX) protocol is an open message standard not controlled by a single entity. It can be adjusted to meet a service requirement of any enterprise and can be used to facilitate information exchange related to secure transactions. The FIX protocol is first used to support securities transactions among consignors in the United States based on direct information flows. With the development over time, the FIX protocol further relates to the service fields of derivatives and other products, including integrated investment, financial derivative products, and foreign exchange transactions.

[0004] Based on the FIX protocol standard, two parties can directly make transactions or can make transactions on a financial service platform (referred to as a service platform below) established by a third party. With the opening of financial markets and the emergence of a large number of small entities, the service platform-based trading mode has attracted more attention from these entities.

[0005] For the service platform-based trading mode, the entity and the service platform should use the FIX protocol of the same version. Currently, the FIX protocol version has been developed from FIX 1.0 to FIX 5.0. The FIX 5.0 version protocol

separates the session layer protocol in the earlier versions from the application layer protocol and further generates two protocol versions: FIXT X.Y and FIX X.Y. In the existing technologies, the service platform cannot request or restrict a protocol version used by the entity. To ensure compatibility of the FIX protocol versions, the service platform needs to set up service environments of all protocol versions, and it separately develops transaction service logic for different protocol versions. With the upgrade of the FIX protocol version and the continuous growth of service types, the costs of deploying the environments on the service platform will be higher and higher.

SUMMARY

[0006] The present disclosure provides a service implementation method, apparatus, and system based on the FIX protocol to resolve a problem of high costs in deploying a FIX service environment on a service platform.

[0007] To resolve the previous problem, according to a first aspect, the present disclosure provides a service implementation method based on the FIX protocol. The method includes the following: a service platform receiving a FIX message sent by a transmitting end, where the FIX message includes different types of service data, and the service data is in the form of a character string; searching a script list for a service script corresponding to the FIX message based on service type information in the service data, where the script list stores service scripts corresponding to different FIX protocol versions, and the service script includes service logic; and executing the service script and processing the service data based on the service logic.

[0008] According to a second aspect, the present disclosure further provides a service implementation method based on the FIX protocol. The method includes the following: encapsulating a FIX packet into a FIX message by a transmitting end, where the FIX message includes different types of service data and does not include service logic corresponding to a detailed FIX protocol version, and the service data is in the form of a character string; and sending the encapsulated FIX message to a service platform, so the service platform processes the service data based on a service script including service logic.

[0009] According to a third aspect, the present disclosure further provides a service implementation apparatus based on the FIX protocol. The apparatus includes the following: a receiving unit, configured to receive a FIX message sent by a

transmitting end, where the FIX message includes different types of service data, and the service data is in the form of a character string; a searching unit, configured to search a script list for a service script corresponding to the FIX message based on service type information in the service data, where the script list stores service scripts corresponding to different FIX protocol versions, and the service script includes service logic; and an execution unit, configured to execute the service script and process the service data based on the service logic.

[0010] According to a fourth aspect, the present disclosure further provides a service implementation apparatus based on the FIX protocol. The apparatus includes the following: a processing unit, configured to encapsulate a FIX packet into a FIX message, where the FIX message includes different types of service data and does not include service logic corresponding to a detailed FIX protocol version, and the service data is in the form of a character string; and a sending unit, configured to send the encapsulated FIX message to a service platform, so the service platform processes the service data based on a service script including service logic.

[0011] According to a fifth aspect, the present disclosure further provides a service implementation system based on the FIX protocol, where the system includes a service platform and a transmitting end; the service platform includes the apparatus according to the third aspect; and the transmitting end includes the apparatus according to the fourth aspect.

[0012] Based on the previous solutions and based on the FIX protocol provided in the present disclosure, the service implementation method, apparatus, and system can perform high-level abstraction and encapsulation on the FIX packet to obtain a universal FIX message that retains only service data, but it does not include detailed service logic. By removing the detailed service logic from the FIX packet, the strongly-typed service of the FIX packet is weakened, so the FIX message can be used for the FIX protocol of any version. The service logic is configured independently of the FIX protocol code by using a service script. Different service scripts are developed for different FIX protocol versions. The FIX transaction service is implemented based on the features of "dynamic configuration" and "dynamic loading" of the script language. Compared with the existing technologies, the present disclosure implements complete decoupling between the service logic and the FIX protocol code, so the implementation of the service logic is no longer dependent on the protocol code itself. Therefore, the service platform can implement transaction

services of all FIX protocol versions by deploying only one version of FIX protocol, so the deployment costs of the FIX service environment can be greatly reduced.

[0013] The previous description is merely an overview of the technical solutions of the present disclosure. To better understand the technical means of the present disclosure to implement the content of the specification, and to make the previous and other objectives, features, and advantages of the present disclosure more comprehensible, detailed implementations of the present disclosure are enumerated as follows.

BRIEF DESCRIPTION OF DRAWINGS

[0014] By reading the detailed descriptions of the following implementations, various other advantages and benefits can be understood by a person of ordinary skill in the art. The accompanying drawings are merely used to illustrate a purpose of an implementation and are not considered as a limitation on the present disclosure. In addition, the same reference symbols are used to represent the same components throughout the drawings. In the accompanying drawings:

[0015] FIG. 1 shows a flowchart illustrating a first service implementation method based on the FIX protocol, according to an implementation of the present disclosure;

[0016] FIG. 2 shows a flowchart illustrating a second service implementation method based on the FIX protocol, according to an implementation of the present disclosure;

[0017] FIG. 3 shows a flowchart illustrating a third service implementation method based on the FIX protocol, according to an implementation of the present disclosure;

[0018] FIG. 4 shows a flowchart illustrating a method for parsing a non-standard FIX protocol, according to an implementation of the present disclosure;

[0019] FIG. 5 shows a first interaction diagram for processing a FIX message, according to an implementation of the present disclosure;

[0020] FIG. 6 shows a second interaction diagram for processing a FIX message, according to an implementation of the present disclosure;

[0021] FIG. 7 shows a composition block diagram illustrating a first service implementation apparatus based on the FIX protocol, according to an implementation of the present disclosure;

[0022] FIG. 8 shows a composition block diagram illustrating a second service implementation apparatus based on the FIX protocol, according to an implementation of the present disclosure;

[0023] FIG. 9 shows a composition block diagram illustrating a third service implementation apparatus based on the FIX protocol, according to an implementation of the present disclosure; and

[0024] FIG. 10 shows a schematic diagram illustrating a service implementation system based on the FIX protocol, according to an implementation of the present disclosure.

DESCRIPTION OF EMBODIMENTS

[0025] The following describes the exemplary implementations of the present disclosure in more detail with reference to the accompanying drawings. Although the accompanying drawings show exemplary implementations of the present disclosure, it should be understood that the present disclosure can be implemented in various forms, and it shall not be limited by the implementations described here. Instead, these implementations are provided to provide a more thorough understanding of the present disclosure and to completely convey the scope of the present disclosure to a person skilled in the art.

[0026] An implementation of the present disclosure discloses a service implementation method based on the Financial Information Exchange (FIX) protocol. As shown in FIG. 1, the method includes:

[0027] 101. A service platform receives a FIX message sent by a transmitting end.

[0028] The transmitting end is usually an entity, such as a client of a stock exchange and a futures exchange. The transmitting end sends an encapsulated FIX message to the service platform, where content of the FIX message is in the form of a character string and includes different types of service data, such as a transaction amount, a transaction time, IDs of a buyer and a seller, a service type, and a message length. In actual application, all data fields (or referred to as fields) in the existing FIX protocol can be used as service data in the present implementation. In the existing FIX protocol, the data fields predetermined by both transaction parties can also be used as the service data. The present implementation sets no limitation on the type and the amount of service data.

[0029] The FIX message received in the present step is a message obtained after the transmitting end performs high-level abstraction and encapsulation. A difference from the existing implementation lies in that in addition to including various data fields; the existing FIX packet includes service logic specific to a detailed transaction service. The service logic is essentially constraint control information for the various data fields. For example, for a service "buying stock", the detailed constraint control information includes at least the following: determining a buying object (such as a stock), determining whether a stock price reaches a predetermined price, buying how many shares, and how to initiate a buying operation. Because the service logic in the FIX packet covers a detailed constraint control condition for implementing a transaction service, a FIX packet is referred to as a strongly-typed packet in an existing technology. In addition, because detailed service logic exists in the FIX packet, for the same transaction service, the service logic corresponding to different FIX protocol versions is different due to the difference among FIX protocol versions. Therefore, the service logic needs to be separately developed for different FIX protocol versions in an existing technology. In the present implementation, the service logic is decoupled from the FIX packet and is separately configured on the service platform in the form of the script language. For the FIX packet, the transmitting end abstracts its data fields (service data) into a universal class to obtain service data in the form of a character string. The universal service data can be applied to any FIX protocol version.

[0030] 102. The service platform searches a script list for a service script corresponding to the FIX message based on service type information in service data.

[0031] Each service script records one piece of detailed service logic corresponding to a detailed FIX version. The script list is an index file used to search for a service script, and records a mapping relationship between service types and service scripts. After receiving the FIX message, the service platform searches the script list for a corresponding service script, based on the service type information, to implement corresponding service logic.

[0032] Further, considering that one piece of service logic can have service scripts corresponding to different FIX protocol versions, for accurate identifying, the service platform can further extract protocol version information from the service data. Moreover, it can combine the protocol version information with the service type information to identify the service script. Correspondingly, one dimension of

information, the protocol version information, should be added to the script list to record a mapping relationship among service types, protocol versions, and service scripts.

[0033] In the present implementation, a developer needs to pre-compile various service scripts, and the service platform can allow the developer to develop different service scripts corresponding to various pieces of service logic for a certain FIX protocol version. It can allow the developer to develop different service scripts corresponding to all FIX protocol versions for one piece of detailed service logic. In actual application, to implement on-demand service logic customization, the service platform can further open an Application Programming Interface (API) for developing service scripts to an entity so as to receive a service script developed by the entity.

[0034] It is worthwhile to note that the script language is a light dynamic language, and the time costs and the computer resources consumed for developing the script language are far less than the consumption of developing the service logic in the FIX protocol code. Therefore, compared with the existing technologies, in addition to reducing the number of deployed protocol versions, the present implementation can reduce deployment costs of the FIX protocol by configuring the script language.

[0035] 103. The service platform executes the service script and processes the service data based on service logic.

[0036] Generally, the service script can be stored on a database server. After identifying the needed service script in the database, the service platform puts it into the cache and loads the service script dynamically by refreshing the cache.

[0037] Generally, service logic implementation can use different service data. For example, in a commodity transaction process, at least the service data such as a buyer ID, a seller ID, a commodity ID, and a commodity price are needed. Selecting which type of service data and how to process the service data depend on the service logic in the service script; in other words, the constraint control information in the service logic. The present implementation does not restrict the type and the number of service data used for executing the service script.

[0038] In actual application, some transaction services are completed on the service platform, while some transaction services need to be transferred by the service platform to an entity that serves as a receiving end (e.g., negotiating service). For the latter case, after step 103 is performed, the service platform further needs to send the processed FIX message to the receiving end.

[0039] Further, an implementation of the present disclosure discloses a service implementation method based on the FIX protocol. As shown in FIG. 2, the method includes:

[0040] 201. A transmitting end encapsulates a FIX packet into a FIX message.

[0041] The method shown in FIG. 2 is mainly applied to the transmitting end, to perform high-level abstraction and encapsulation on a conventional FIX packet to obtain a universal FIX message. Because the FIX message includes only service data in the form of a character string, and does not include service logic corresponding to a detailed FIX protocol version, the FIX message can be applied to a FIX protocol of any version.

[0042] In addition, the service logic is written into a service script developed in advance. It is configured on a service platform, so the service platform processes the service data based on the service script that includes the service logic.

[0043] 202. The transmitting end sends the encapsulated FIX message to a service platform.

[0044] In the method provided in the present implementation of the present disclosure, the service logic in the FIX packet can be reduced to a service script on the service platform for implementation. Constraint control information in the service logic is different in form, and the content based on different FIX protocol versions. Therefore, when the FIX message includes universal service data and does not include detailed service logic, the FIX message is not restricted by the FIX protocol version, and can be applied to the FIX protocol of any version. Compared with the existing technologies, the present implementation can improve the compatibility of the FIX protocol versions, reduce the number of deployed FIX protocol versions, and thus reduce the deployment costs of the FIX service environment.

[0045] Further, as an improvement of the methods shown in FIG. 1 and FIG. 2, an implementation of the present disclosure further provides a service implementation method based on the FIX protocol. As shown in FIG. 3, the method includes:

[0046] 301. A transmitting end encapsulates a FIX packet into a FIX message.

[0047] The transmitting end encapsulates the FIX packet into a universal FIX message. The transmitting end extracts character strings from the header and the body of the FIX packet to assemble a universal FIX message. The transmitting end captures the packet header based on a predetermined header length, parses out a body length based on header content, captures the character string of the body based on the body

length, and finally assembles the captured character strings into the FIX message.

[0048] In actual application, the transmitting end can, but is not limited to, encapsulating the FIX packet into a message-type FIX message.

[0049] 302. The transmitting end sends the encapsulated FIX message to a service platform.

[0050] Based on the existing FIX protocol, both receiving and sending parties of the FIX packet should communicate with each other in one session. After the session is established, the transmitting end invokes the existing sending method to send the FIX message to the service platform, for example, send the FIX message by using the static method sendToTarget.

[0051] 303. The service platform searches a script list for a service script corresponding to the FIX message based on service type information in service data.

[0052] As described above, both the message receiving and sending parties need to communicate with each other based on the same session. When sending the FIX message, the transmitting end sends the current session ID together. The service platform identifies a corresponding session in a session list based on the session ID, and then performs a subsequent procedure in the session.

[0053] In the existing technologies, the service data in the FIX packet exists in the form of a data field; namely, exists in the form of a tag-value pair. The service data in the present implementation is also present in the form of a tag-value pair. The following represents some tag-value pairs specified by the FIX protocol:

Tag	**Field Name**	**Description**	**Must be sent**	**Remarks**
8	BeginString	Protocol version number	Y	FIX.4.2 by default, cannot be encrypted, and must be the first field of the message
9	BodyLength	Body length	Y	Cannot be encrypted and must be the second field of the message, excluding the length of 8/9/10 field
35	MsgType	Message type	Y	Cannot be encrypted and must be the third field of the message
49	SenderCompID	Sender ID	Y	Cannot be encrypted

56	TargetCompID	Receiver ID	Y	Cannot be encrypted
34	MsgSeqNum	Session sequence number	Y	Resetting is not allowed during transaction unless it is the first login in one day.
52	SendingTime	Sending time	Y	In the worldwide uniform time format

[0054] After parsing the FIX message, the transmitting end obtains the service data included in the message. One tag-value pair corresponds to one type of service data.

[0055] After the tag-value pair is obtained, the service platform searches for the tag-value pair of the service type, obtains service type information of a transaction service, and searches the script list for the service script corresponding to the FIX message based on the service type information.

[0056] Further, because the service platform involves numerous transaction services the service platform can further search for a service script in combination with a sender ID, a message type (an application layer message or a session layer message), and even a session ID to avoid repetition or confusion.

[0057] 304. The service platform executes the service script and processes the service data based on service logic.

[0058] The service platform puts the identified service script into the cache, and loads the service script by refreshing the cache to implement corresponding service logic.

[0059] In actual application, the service logic related to the FIX protocol can be divided into two categories:

[0060] One category is service logic specific to detailed transaction objects or detailed transactions. The service logic cannot be universally used in different transaction objects or different transactions, and it needs to be separately developed for detailed cases, including, but not limited to, packet content parsing (related to different parsing rules), packet field check (such as the data format, amount format, amount, etc.), packet field conversion (date format conversion, amount format conversion, currency identifier conversion), signature verification, return code conversion, idempotent control, and traffic control.

[0061] The other category is service logic that can be used by all entities. The

present type of service logic is mainly used to establish, guarantee, and deregister the bottom-layer communication process between two communication parties. It also includes more universal service logic of transaction services, which is applicable to any service type. Typical service logic includes heartbeat management, network test, message retransmission, message rejection, sequence number reset, deregistration, login verification at the session layer, withdrawal request, rejection order, execution report, status request, status check, and new order at the application layer.

[0062] Particular service logic is configured on the service platform in the form of a service script, which is implemented by performing step 304. However, for universal service logic, the service script does not need to be repeatedly configured for different entities. In developing the FIX service environment, the universal service logic is directly written into the protocol code on the service platform, which is implemented by performing subsequent steps 305 to 307. Since the universal service logic is applicable to all service types and does not need to be modified during usage, the maintenance costs of the FIX service environment are not increased.

[0063] 305. The service platform determines a message type of the FIX message based on message type information in the service data.

[0064] The service platform obtains the message type of the FIX message based on the tag-value pair with key "MsgType". The message type includes two types: application layer message and session layer message. For different message types, the service platform chooses to perform subsequent step 306 or 307.

[0065] 306. If the message type indicates an application layer message, the service platform invokes an application layer method to execute universal service logic at an application layer.

[0066] The service platform invokes the toApp method or the fromApp method in the Application class to execute the universal service logic at the application layer. The difference between the toApp method and the fromApp method lies in following conditions: If a FIX message sent to the receiving end is processed, the toApp method is invoked. If a received FIX message is processed, the fromApp method is invoked.

[0067] 307. If the message type indicates a session layer message, the service platform invokes a session layer method to execute universal service logic at a session layer.

[0068] The service platform invokes the toAdmin method or the fromAdmin method in the Application class to execute the universal service logic at the

application layer. The difference between the toAdmin method and the fromAdmin method lies in following conditions: If a FIX message sent to the receiving end is processed, the toAdmin method is invoked. If a received FIX message is processed, the fromAdmin method is invoked.

[0069] In actual application, the session sequence numbers of sent and received FIX messages are separately maintained. The message queues of sent and received messages are also mutually independent. Therefore, the service platform can choose to invoke toApp/toAdmin or fromApp/fromAdmin based on the session sequence numbers of the FIX messages.

[0070] In the present implementation, step 305 to step 307 following step 304 is merely used for ease of description. In actual application, after receiving the FIX message, the service platform can first perform step 305 to step 307 and then step 303 to step 304, or it can perform step 305 to step 307 and step 303 to step 304 simultaneously.

[0071] In an implementation of the present implementation, universal check services, such as field check and transmission loss check, based on session sequence number can be written into the FIX protocol code and implemented by invoking a session layer method.

[0072] Further, the service platform can further write the following new types of universal service logic into the FIX protocol code and execute the service logic by using the session layer method:

[0073] 1. Reset sequence number logic

[0074] The existing FIX protocol standard contains a sequence number protection mechanism. The message sender adds a session sequence number to the sent message. For the message receiver, if the sequence number of the currently received message and the sequence number of the previously received message are not consecutive, it indicates that there is a missed message between the two messages. The message retransmission mechanism needs to be triggered. If the sequence numbers are consecutive, the sender sends a message again, and the sequence number of the previous message is increased by 1, which is used as the session sequence number of the next message.

[0075] It can be seen that with the continuous development of the communication process, session sequence numbers used by both communication parties will increase. When the value of the session sequence number exceeds the upper limit supported by

the system or the database, error retransmission can occur easily because the system cannot identify the session sequence number. To prevent the session sequence number from exceeding the system upper limit, the present implementation provides universal service logic for resetting a sequence number, which is implemented by using the session layer method. The service platform sets a predetermined maximum sequence number based on the upper limit supported by the system. Generally, the maximum value of the sequence number is less than the upper limit of the system. In actual application, the maximum value of the sequence number can be set to 70% to 95% of the upper limit of the system. For example, when the upper limit of the system is 990,000, the maximum value of the sequence number can be set to 900,000.

[0076] When resetting sequence number logic, the service platform determines whether the session sequence number of the current message reaches the predetermined maximum sequence number. If the maximum sequence number is reached, the session sequence number is reset to 0, so the subsequent sequence number is accumulated starting from 0. If the maximum sequence number is not reached, no processing is performed.

[0077] In the present implementation, the reset sequence number logic of the session layer can prevent a session sequence number of a message from exceeding the upper limit supported, by the system or the database, to effectively avoid a system exception caused by a session sequence number overflow.

[0078] 2. Refuse transaction logic

[0079] When the session sequence number is reset, a reset failure can be caused because the entity refuses to process or because of a system exception. In this case, to prevent the session sequence number from exceeding the upper limit supported by the system or the database, the service platform can implement the refuse transaction logic by using the session layer method, and refuse to process the current FIX message so as to eliminate the possibility of further accumulating the session sequence number.

[0080] 3. Clear queue logic

[0081] In a retransmission mechanism of the existing FIX protocol, if session sequence numbers are not consecutive, the current message is cached in a message queue, and the transmitting end is instructed to re-transmit a message corresponding to a missed sequence number. In this case, all messages following the current message sent by the transmitting end need to be cached in the message queue, and there is a

memory overflow risk in the message queue. In the present implementation, the service platform can monitor a size of the message queue by performing the clear queue logic. If a cached occupied by the message queue exceeds a predetermined cache threshold, the FIX messages cached in the message queue are cleared to ensure that the system is stable and highly available. In actual application, the cache threshold can be proportionally set based on a memory size of the system, and a detailed value is not limited in the present implementation.

[0082] Further, in the present implementation, when the entity adds a FIX service, the service platform can further obtain a service script that includes the service logic of the newly added service, and add the obtained service script to the script list. In the present implementation, the service logic is configured on the service platform in the form of the script language. When service logic needs to be added, only the newly developed service script needs to be configured on the service platform, and the FIX protocol code does not need to be modified. Compared with the existing technologies, the flexibility and convenience of adding service logic are greatly improved, which makes it easier for the entity to customize personalized transaction services.

[0083] Similarly, when some service logic needs to be deleted from the service platform, only corresponding service scripts need to be identified based on the service types, and then the service scripts are deleted from the script list. Likewise, the FIX protocol code does not need to be changed.

[0084] Further, considering that in actual application, some entities do not use the standard FIX protocol packet for communication because they do not want to repeatedly reset the session sequence number. For an entity that uses a non-standard FIX protocol packet, the service platform can further pre-develop different packet parsers based on encapsulation rules of different non-FIX protocol packets in the present implementation to implement its transaction services. When a non-FIX protocol packet is received, the service platform selects a corresponding packet parser based on the encapsulation rule, and parses the data stream of the non-FIX protocol packet by using the packet parser to obtain service data.

[0085] In actual application, the assets query transaction of the securities firms is implemented by using non-standard FIX protocol packets. The non-standard FIX protocol is used to reduce the increasing speed of sequence numbers. Because the frequency of assets query is significantly higher than the transaction frequency, the assets query will inevitably result in the rapid growth of the session sequence numbers,

which will result in frequent resetting. For the non-standard FIX protocol packets, because their encapsulation rules are different from the encapsulation rules of the universal FIX message, the universal parsing means cannot correctly parse the underlying binary data stream into data in the form of a character string. In the present implementation, correctly parsing the non-standard FIX protocol packets can be implemented by using a customized packet parser, which ensures implementation of the service logic. A process of parsing the non-standard FIX protocol packet by using the packet parser is shown in FIG. 4.

[0086] 401. Capture a packet header based on an agreed header length.

[0087] The service platform captures the packet header from the data stream of the non-standard FIX protocol packet based on the header length specified in the encapsulation rule. The header length can be agreed upon by the service platform in advance with the entity (transmitting end).

[0088] 402. Obtain a packet type and body length information based on the packet header.

[0089] After the packet header is obtained, the field information such as a packet type and a body length can be obtained. The service platform performs step 403 based on the body length information to capture the body.

[0090] 403. Capture a body based on the body length.

[0091] In a process of capturing the body, the service platform needs to determine whether the captured body is long enough on a plurality of occasions, in other words, whether the captured body reaches the body length. If the length reaches the body length, the capturing is stopped. Otherwise, the data stream continues to be read until the body length is reached or the data stream is completely read.

[0092] 404. Assemble the packet header and the body and return it to an upper layer in the form of a character string.

[0093] After the data stream corresponding to the packet header and the body is captured, the data stream is assembled into data in the form of a character string, and is returned to the upper layer. The upper layer splits the character string into tag-value pairs. After the field check, the extraction of the service data is completed.

[0094] Further, in another implementation of the present implementation, after processing the received FIX message, the service platform can further send the processed FIX message to the receiving end. Generally, in addition to transaction services that can be implemented on the service platform, there are some transaction

services that need to be implemented jointly by both parties, for example, price inquiry/counter-offer. In this case, the service platform needs to process the FIX message sent by one party (transmitting end), and send the processed FIX message to the other party (receiving end). The implementation of the present disclosure subsequently describes the implementation processes of the FIX service based on two processes: message receiving and message sending.

[0095] Further, in the last implementation of the present implementation, the service platform can further use a distributed network architecture to perform distributed processing on the received FIX message by using a plurality of computing nodes. Each computing node processes a FIX message of at least one session. In an initial session phase, load balancing can be performed on a session based on a session ID by using a hash algorithm, so each computing node is responsible for a same or approximately same number of sessions. When a session is newly added, the session can be allocated to a computing node with the least number of sessions, or a newly added session can be allocated to a computing node that has recently performed session deregistration when the computing nodes have the same number of sessions. The present implementation sets no limitation on the number of computing nodes, a session allocation method, and the number of sessions on a node in the distributed structure.

[0096] In the previous implementation, in addition to improving computing efficiency and reducing load of a single computing node, the distributed network structure enables each computing node to independently manage its session sequence numbers to support horizontal extension of session sequence numbers, thereby resolving a performance bottleneck caused by sequence number sequential sending.

[0097] The following provides two application scenarios of the implementation of the present disclosure.

[0098] First, the service platform sends the FIX message to the outside. In this scenario, the service platform receives the FIX message sent by the transmitting end, and processes the FIX message and sends the processed FIX message to the receiving end. As shown in FIG. 5, the scenario procedure includes the following steps.

[0099] 1. The transmitting end encapsulates a FIX message, and invokes the static method SendToTarget to send the FIX message to the service platform.

[0100] 2. The service platform searches for a corresponding session based on a session ID included in the message.

[0101] 3. The service platform searches for and executes a corresponding groovy script based on service type information included in the message. The FIX message is processed by using the groovy script to implement the service logic in the script.

[0102] 4. The service platform determines a type of the FIX message based on message type information included in the message. If the type indicates a session layer message, the toAdmin method at the session layer is invoked to process the FIX protocol to implement the universal service logic of the session layer. If the type indicates an application layer message, the toApp method at the application layer is invoked to process the FIX protocol to implement the universal service logic of the application layer.

[0103] 5. The service platform invokes object IoSessionResponder to send the processed FIX message to the receiving end. Object IoSessionResponder is initialized by InitiatorIOHandler.

[0104] 6. The service platform and the transmitting end increase respective stored session sequence numbers by 1.

[0105] Second, the service platform receives the FIX message sent by the outside. In this scenario, the service platform receives the FIX message sent by the transmitting end, and processes the FIX message. As shown in FIG. 6, the scenario procedure includes the following steps.

[0106] 1. The service platform performs asynchronous non-blocking monitoring on the socket of the transmitting end by using object InitiatorIoHandler.

[0107] 2. When receiving the FIX message sent by the transmitting end, the service platform invokes class EventHandlingStrategy to add the message to message queue blockingQueue.

[0108] 3. The service platform invokes class EventHandlingStrategy to repeatedly read the FIX message from the blockingQueue.

[0109] 4. The service platform searches for a corresponding session based on a session ID included in the message.

[0110] 5. The service platform checks whether session sequence numbers are consecutive. If the session sequence numbers are consecutive, perform subsequent steps. Otherwise, the transmitting end resends the FIX message within the normal sequence number range.

[0111] 6. The service platform searches for and executes a corresponding groovy script based on service type information included in the message. The FIX message is

processed by using the groovy script to implement the service logic in the script.

[0112] 7. The service platform determines a type of the FIX message based on message type information included in the message. If the type indicates a session layer message, the fromAdmin method at the session layer is invoked to process the FIX protocol to implement the universal service logic of the session layer. If the type indicates an application layer message, the fromApp method at the application layer is invoked to process the FIX protocol to implement the universal service logic of the application layer.

[0113] 8. The service platform and the transmitting end increase respective stored session sequence numbers by 1.

[0114] In the previous scenarios, the service script is developed by using the groovy language. Groovy is an agile dynamic language based on the Java virtual machine. It can seamlessly integrate all existing Java objects and class libraries. It is more convenient to use the groovy language and the Java environment to develop the service script. In actual application, the service script can further be implemented by using languages based on the Java virtual machine, such as JPython, JRuby, etc., which is not limited in the present implementation.

[0115] Further, as an implementation of the methods shown in FIG. 1 and FIG. 3, an implementation of the present disclosure further provides a service implementation apparatus based on the FIX protocol. The apparatus is mainly located on a service platform. As shown in FIG. 7, the apparatus includes a receiving unit 71, a searching unit 72, and an execution unit 73.

[0116] The receiving unit 71 is configured to receive a FIX message sent by a transmitting end, where the FIX message includes different types of service data, and the service data is in the form of a character string. The searching unit 72 is configured to search a script list for a service script corresponding to the FIX message based on service type information in the service data, where the script list stores service scripts corresponding to different FIX protocol versions. The service script includes service logic, and the execution unit 73 is configured to execute the service script and process the service data based on the service logic.

[0117] Further, the FIX message received by the receiving unit 71 is a universal FIX message.

[0118] Further, as shown in FIG. 8, the apparatus further includes a write unit 74, configured to write universal service logic into protocol code on the service platform,

where the universal service logic is applicable to any service type.

[0119] Further, as shown in FIG. 8, the apparatus further includes a determining unit 75, configured to determine a message type of the FIX message based on message type information in the service data after the FIX message sent by the transmitting end is received.

[0120] The execution unit 73 is configured to the following: when the message type indicates an application layer message, invoke an application layer method to execute universal service logic at an application layer; and the execution unit 73 is further configured to invoke a session layer method to execute universal service logic at a session layer when the message type indicates a session layer message.

[0121] Further, the execution unit 73 is configured to the following: when the universal service logic is a reset sequence number logic at the session layer, if a session sequence number reaches a predetermined largest sequence number, reset the session sequence number to 0; or when the universal service logic is a refuse transaction logic at the session layer, if resetting a session sequence number fails, refuse to process the FIX message; or when the universal service logic is clear queue logic at the session layer, if a cache occupied by a message queue exceeds a predetermined cache threshold, clear a cached FIX message in the message queue.

[0122] Further, as shown in FIG. 8, the apparatus further includes the following: an acquisition unit 76, configured to obtain a service script including service logic of the added FIX service when a FIX service is added; and an adding unit 77, configured to add the obtained service script to the script list.

[0123] Further, as shown in FIG. 8, the apparatus further includes the following: a selection unit 78, configured to select a packet parser based on an encapsulation rule of the non-FIX-protocol-based packet when a non-FIX-protocol-based packet; and a parsing unit 79, configured to parse a data stream of the non-FIX-protocol-based packet by using the packet parser to obtain service data.

[0124] Further, as shown in FIG. 8, the apparatus further includes a sending unit 710, configured to send a processed FIX message to a receiving end.

[0125] Further, the apparatus is configured to perform distributed processing on a received FIX message by using a plurality of computing nodes, where each computing node processes a FIX message of at least one session.

[0126] Further, as an implementation of the methods shown in FIG. 2 and FIG. 3, an implementation of the present disclosure further provides a service implementation

apparatus based on the FIX protocol. The apparatus is mainly located on a transmitting end. As shown in FIG. 9, the apparatus includes a processing unit 91 and a sending unit 92.

[0127] The processing unit 91 is configured to encapsulate a FIX packet into a FIX message, where the FIX message includes different types of service data and does not include service logic corresponding to a detailed FIX protocol version. The service data is in the form of a character string, and the sending unit 92 is configured to send the encapsulated FIX message to a service platform, so the service platform processes the service data based on a service script including service logic.

[0128] Further, the processing unit 91 is configured to encapsulate a FIX packet into a universal FIX message.

[0129] Further, the processing unit 91 is configured to extract character strings from the header and the body of the FIX packet to assemble a universal FIX message.

[0130] Further, as an implementation of the method shown in FIG. 3, an implementation of the present disclosure further provides a service implementation system based on the FIX protocol. As shown in FIG. 10, the system includes a service platform 101 and a transmitting end 102.

[0131] The service platform 101 includes the apparatus shown in FIG. 7 or FIG. 8, and the transmitting end 102 includes the apparatus shown in FIG. 9.

[0132] The service implementation apparatus and system based on the FIX protocol provided in the implementations of the present disclosure can perform high-level abstraction and encapsulation on the FIX packet, to obtain a universal FIX message that retains only service data, but does not include detailed service logic. By removing the detailed service logic from the FIX packet, the strongly-typed service of the FIX packet is weakened, so the FIX message can be used for the FIX protocol of any version. The service logic is configured independently of the FIX protocol code by using a service script. Different service scripts are developed for different FIX protocol versions. The FIX transaction service is implemented based on the features of "dynamic configuration" and "dynamic loading" of the script language. Compared with the existing technologies, the implementation of the present disclosure implements complete decoupling between the service logic and the FIX protocol code, so the implementation of the service logic is no longer dependent on the protocol code itself. Therefore, the service platform can implement transaction services of all FIX protocol versions by deploying only one version of FIX protocol, so the deployment

costs of the FIX service environment can be greatly reduced.

[0133] In the previous implementations, the description of each implementation has respective focuses. For a part not described in detail in an implementation, reference can be made to related descriptions in other implementations.

[0134] It can be understood that related features in the previous method and apparatus can be mutually referenced. In addition, "first", "second", etc. in the previous implementation are used to distinguish between implementations, and do not represent advantages and disadvantages of each implementation.

[0135] A person skilled in the art can clearly understand that for the convenience and brevity of the description, for a specific working process of the system, the apparatus, and the unit described above, reference can be made to a corresponding process in the previous method implementations, and details are not described here again.

[0136] The algorithm and display provided here are not inherently related to any particular computer, virtual system, or other devices. Various general-purpose systems can also be used together with the instruction based here. Based on the above description, the structure needed for constructing such a system is obvious. Furthermore, the present disclosure does not apply to any particular programming language. It should be understood that the content of the present disclosure described here can be implemented by using various programming languages, and the previous description of a detailed language is used to disclose the best implementation of the present disclosure.

[0137] A large number of details are provided in the specification. However, it can be understood that the implementations of the present disclosure can be practiced without these details. In some instances, well-known methods, structures, and technologies are not shown in detail, so as not to blur the understanding of the present specification.

[0138] Similarly, it should be understood that, to streamline the present disclosure and help understand one or more of various inventive aspects, the features of the present disclosure are sometimes grouped together into a single implementation, drawing, or description of the present disclosure. However, the disclosed method should not be interpreted as reflecting the following intention: In other words, the claimed disclosure requires more features than the features specified in each claim. More accurately, as reflected in the claims below, the aspects of disclosure are less

than all features of a single implementation previously disclosed. Therefore, the claims that follow a detailed implementation definitely incorporate the detailed implementation. Each claim serves as a separate implementation of the present disclosure.

[0139] A person skilled in the art can understand that modules in the devices in the implementations can be adaptively changed and disposed in one or more devices different from the implementation. Modules, units, or components in the implementations can be combined into one module, unit, or component and can additionally be divided into a plurality of submodules or subunits or subcomponents. Except a fact that at least some of these features and/or processes or units are mutually exclusive, all disclosed features and all processes or units of any method or device that are disclosed in such a way in the specification (including the appended claims, the abstract, and the accompanying drawings) may be combined in any combination mode. Unless otherwise explicitly stated, each feature disclosed in the specification (including the appended claims, the abstract, and the accompanying drawings) may be replaced by an alternative feature that serves same, equivalent, or similar purposes.

[0140] In addition, a person skilled in the art can understand that, although some implementations described here include some features included in another implementation instead of including another feature, a combination of features of different implementations means falling within the scope of the present disclosure and forming different implementations. For example, in the following claims, any one of the contemplated implementations may be used in any combination mode.

[0141] Implementations of various parts in the present disclosure may be implemented by hardware or software modules running on one or more processors, or a combination thereof. A person skilled in the art should understand that a microprocessor or a digital signal processor (DSP) may be used in practice to implement some or all of the functions of some or all of the components of the disclosure name (for example, an apparatus for determining a link level in a website) based on an implementation of the present disclosure. The present disclosure may also be implemented as a device or apparatus program (e.g., computer program and computer program product) for executing a part or all of the methods described here. Such a program for implementing the present disclosure may be stored in a computer-readable medium, or may have a form of one or more signals. Such a signal

may be downloaded from an Internet website, provided on a carrier signal, or provided in any other form.

[0142] It should be noted that the foregoing implementations are intended for describing the present disclosure, instead of limiting the present disclosure, and a person skilled in the art may design an alternative implementation without departing from the scope of the appended claims. In the claims, any reference symbol located between brackets should not be constructed as a limitation on the claims. The word "include" does not exclude the existence of elements or steps not listed in the claims. The word "one" or "a/an" before the element does not exclude the presence of a plurality of such elements. The present disclosure may be implemented by hardware including several different elements and a computer that is appropriately programmed. In the unit claims enumerating several apparatus, several of these apparatus may be embodied by using the same hardware item. The use of words first, second, and third does not indicate any sequence. These words may be interpreted as names.

CLAIMS

What is claimed is:

1. A service implementation method based on the financial information exchange FIX protocol, comprising:

receiving, by a service platform, a FIX message sent by a transmitting end, wherein the FIX message comprises different types of service data, and the service data is in the form of a character string;

searching a script list for a service script corresponding to the FIX message based on service type information in the service data, wherein the script list stores service scripts corresponding to different FIX protocol versions, and the service script comprises service logic; and

executing the service script and processing the service data based on the service logic.

2. The method according to claim 1, wherein the FIX message is encapsulated into a universal FIX message.

3. The method according to claim 1, wherein the method further comprises:

writing universal service logic into protocol code on the service platform, wherein the universal service logic is applicable to any service type.

4. The method according to claim 3, wherein after the receiving a FIX message sent by a transmitting end, the method further comprises:

determining a message type of the FIX message based on message type information in the service data;

if the message type indicates an application layer message, invoking an application layer method to execute universal service logic at an application layer; and

if the message type indicates a session layer message, invoking a session layer method to execute universal service logic at a session layer.

5. The method according to claim 4, wherein the invoking a session layer method to execute universal service logic at a session layer comprises:

when the universal service logic is reset sequence number logic at the session layer, if a session sequence number reaches a predetermined largest sequence number, resetting the session sequence number to 0; or

when the universal service logic is refuse transaction logic at the session layer, if

resetting a session sequence number fails, refusing to process the FIX message; or

when the universal service logic is clear queue logic at the session layer, if a cache occupied by a message queue exceeds a predetermined cache threshold, clearing a cached FIX message in the message queue.

6. The method according to claim 1, wherein the method further comprises:

when a FIX service is added, obtaining a service script comprising service logic of the added FIX service; and

adding the obtained service script to the script list.

7. The method according to claim 1, wherein the method further comprises:

when a non-FIX-protocol-based packet, selecting a packet parser based on an encapsulation rule of the non-FIX-protocol-based packet; and

parsing a data stream of the non-FIX-protocol-based packet by using the packet parser to obtain service data.

8. The method according to claim 1, wherein the method further comprises:

sending a processed FIX message to a receiving end.

9. The method according to any one of claims 1 to 8, wherein the method further comprises:

performing distributed processing on a received FIX message by using a plurality of computing nodes, wherein each computing node processes a FIX message of at least one session.

10. A service implementation method based on the financial information exchange FIX protocol, comprising:

encapsulating, by a transmitting end, a FIX packet into a FIX message, wherein the FIX message comprises different types of service data and does not comprise service logic corresponding to a detailed FIX protocol version, and the service data is in the form of the character string; and

sending the encapsulated FIX message to a service platform, so the service platform processes the service data based on a service script comprising service logic.

11. The method according to claim 10, wherein the encapsulating a FIX packet into a FIX message comprises:

encapsulating the FIX packet into a universal FIX message.

12. The method according to claim 11, wherein the encapsulating the FIX packet into a universal FIX message comprises:

extracting character strings from the header and the body of the FIX packet to

assemble a universal FIX message.

13. A service implementation apparatus based on the financial information exchange FIX protocol, comprising:

a receiving unit, configured to receive a FIX message sent by a transmitting end, wherein the FIX message comprises different types of service data, and the service data is in the form of the character string;

a searching unit, configured to search a script list for a service script corresponding to the FIX message based on service type information in the service data, wherein the script list stores service scripts corresponding to different FIX protocol versions, and the service script comprises service logic; and

an execution unit, configured to execute the service script and process the service data based on the service logic.

14. The apparatus according to claim 13, wherein the FIX message received by the receiving unit is encapsulated into a universal FIX message.

15. The apparatus according to claim 13, wherein the apparatus further comprises:

a write unit, configured to write universal service logic into protocol code on a service platform, wherein the universal service logic is applicable to any service type.

16. The apparatus according to claim 15, wherein the apparatus further comprises a determining unit, configured to determine a message type of the FIX message based on message type information in the service data after the FIX message sent by the transmitting end is received;

the execution unit is configured to invoke an application layer method to execute universal service logic at an application layer when the message type indicates an application layer message; and

the execution unit is further configured to invoke a session layer method to execute universal service logic at a session layer when the message type indicates a session layer message.

17. The apparatus according to claim 16, wherein the execution unit is configured to: when the universal service logic is reset sequence number logic at the session layer, if a session sequence number reaches a predetermined largest sequence number, reset the session sequence number to 0; or

when the universal service logic is refuse transaction logic at the session layer, if resetting a session sequence number fails, refuse to process the FIX message; or

when the universal service logic is clear queue logic at the session layer, if a cache occupied by a message queue exceeds a predetermined cache threshold, clear a cached FIX message in the message queue.

18. The apparatus according to claim 13, wherein the apparatus further comprises:

an acquisition unit, configured to obtain a service script comprising service logic of the added FIX service when a FIX service is added; and

an adding unit, configured to add the obtained service script to the script list.

19. The apparatus according to claim 13, wherein the apparatus further comprises:

a selection unit, configured to select a packet parser based on an encapsulation rule of the non-FIX-protocol-based packet when a non-FIX-protocol-based packet is received; and

a parsing unit, configured to parse a data stream of the non-FIX-protocol-based packet by using the packet parser to obtain service data.

20. The apparatus according to claim 13, wherein the apparatus further comprises:

a sending unit, configured to send a processed FIX message to a receiving end.

21. The apparatus according to any one of claims 13 to 20, wherein the apparatus is configured to perform distributed processing on the received FIX message by using a plurality of computing nodes, wherein each computing node processes a FIX message of at least one session.

22. A service implementation apparatus based on the financial information exchange FIX protocol, comprising:

a processing unit, configured to encapsulate a FIX packet into a FIX message, wherein the FIX message comprises different types of service data and does not comprise service logic corresponding to a detailed FIX protocol version, and the service data is in the form of the character string; and

a sending unit, configured to send the encapsulated FIX message to a service platform, so the service platform processes the service data based on a service script comprising service logic.

23. The apparatus according to claim 22, wherein the processing unit is configured to encapsulate the FIX packet into a universal FIX message.

24. The apparatus according to claim 23, wherein the processing unit is

configured to extract character strings from the header and the body of the FIX packet to assemble a universal FIX message.

25. A service implementation system based on the financial information exchange FIX protocol, wherein the system comprises a service platform and a transmitting end;

the service platform comprises the apparatus according to any one of claims 13 to 21; and

the transmitting end comprises the apparatus according to any one of claims 22 to 24.