Attachment II – Architecture Mapping of Ardor

Section 1 Summary

Platform summary	
Platform ID	*Ardor / ARDR*
Status/Revision	*Stable Release V2.2.5*
Type	*Public with hybrid capabilities*
Domain	*Blockchain Infrastructure, Payments, Data security*
Description	*Ardor is a multi-chain platform where Ardor, as the parent chain, provides energy-efficient proof of stake consensus for transactions across each of the platform's child chains. All child chains have their own native token and are inherently compatible with one another. Hybrid capabilities enable child chains to be either permissioned or permissionless while leveraging the public permissionless Ardor network for consensus.*

Section 2 Governance & Compliance Functions

Platform governance	
Governance Type	*Permissionless*
Chain Network Admin	*Community (Jelurida, a Swiss based company, as developer)*
Pledge (cost of malicious action)	*Stake for parent chain. Business agreement for child chain.*
Description	*When malicious activity is detected, Jelurida releases an additional version of the software under the Jelurida Public License (JPL) with the necessary security updates. Jelurida then allows the platform stake holders to choose which version represents the legitimate version of the ledger. By doing this, final decisions on how to respond to malicious activity are controlled in a decentralized manner.* *When a company wishes to launch a new child chain within the Ardor ecosystem, they must engage in a business agreement with Jelurida. Jelurida then releases an official upgrade of the Ardor software including the new child chain.*

Platform trust endorsement policy	
Type	*Tokenomics*
Tool	*ARDR*

Policy	*ARDR as stake. Child chain tokens as transaction fees. Bundlers exchange between. New child chains can only be added through a business agreement with Jelurida.* *Trust is accomplished by means of decentralization. The process of staking is called 'forging' on the Ardor network. Forging ARDR is a public process open to anyone with a balance of at least 1000 ARDR. Jelurida as the development team does not exert censorship or transaction reversal capabilities of any kind on chain.*

Economic Model (optional)	
Price Model to Deploy Contracts and do Transactions	*Contracts are deployed as Java data files or jar files to the blockchain using a data cloud transaction type.* *Charged by transactions only.*
Who pays the costs of the network	*Users and/or Businesses* *Users pay fees when interacting directly with the blockchain.* *Businesses have the ability to provide app end-users with 0 fee transactions by sponsoring child chain transaction fees using custom bundlers.* *For info:* https://ardordocs.jelurida.com/Tutorial_on_custom_bundlers_for_child_chain_transactions
Monetary Policy of Tokens	*The supply of tokens on the Ardor parent chain is finite with all 998,999,495 ARDR tokens in circulation at the time of the network's launch. There are options for adjusting the supply of assets and monetary supplies issued on specific child chains.*
Rights of Tokens	*ARDR tokens represent an entity's stake for participating in the process of "forging" with the proof of stake consensus mechanism.*

Section 3 Application

Platform Smart Contract mechanism	
Language	*Java*
Turing Complete?	*Yes*
Compiler	*Java*
Runtime VM	*JVM*
DevTools	*Any Java IDE – software includes built in support for deployment, unit testing, and debugging using IDE plug ins (IntelliJ recommended).*

	1. https://ardordocs.jelurida.com/Lightweight_Contracts 2. https://medium.com/@lyaffe/lightweight-contracts-articles-49c3032a50da 3. https://ardordocs.jelurida.com/Transaction_Vouchers
Extra Tool(s)	*Extensive set of development tools, block explorers and wallets provided out of the box and through 3rd party services.* *Block explorer 1: https://ardor.tools* *Block explorer 2: https://ardorportal.org* *Block explorer 3: https://ardor.world* *Load test:* https://www.jelurida.com/ardor-loadtest-report
Lifecycle	*Contracts are executed by specific events or when they receive a trigger transaction referencing the ID of the node responsible for running that contract. This makes it easy to upgrade, replace, and shutdown contracts. To upgrade or replace a contract, simply deploy the new contract to a node and update the reference ID in future trigger transactions. To shutdown a contract, ensure the node responsible for executing said contract is taken offline.* *https://ardordocs.jelurida.com/Lightweight_Contracts#Contract_Development* *Jelurida, the development company behind Ardor, manages enhancement proposals and formulates an extensive product roadmap.* *https://www.jelurida.com/ardor-roadmap*
Description	*Lightweight smart contracts on Ardor are stateless. Contracts are stored on chain as Java class or jar file but their execution is not part of the PoS consensus. Contracts are typically executed only by a subset of the nodes. Trust is achieved using multi-signature setup of verification and approval nodes. External oracles are allowed. See https://ardordocs.jelurida.com/Lightweight_Contracts and https://medium.com/@lyaffe/lightweight-contracts-articles-49c3032a50da*

Section 4 Protocol

Platform AAA Management	
Account type	*Address*
Distributed ID	*Ardor Account ID*
AAA support	*Full Support*
Description	*256 bit Private/Public key pair is derived from a passphrase (seed) with 128 bits of entropy. Account address is derived from the first 64 bits of the*

	public key SHA256 hash. Reed Solomon error correction is used to generate the public account address. *Authentication is supported using an account passphrase from which the account public key and id are derived* *For the permissioned version only an account authorization level is determined using on chain messages set from a root trust introduced by the Genesis block and standard role-based authentication.* *Account can be placed under account control of possibly one or more other accounts to confirm or reject every transaction submitted by this account.*

Platform Consensus Mechanism	
Algorithm	*PoS*
Consensus mode	*Event*
Management solution	*Internal*
Description	*Chance of generating the next block depends on your account balance i.e. stake.* *Pseudo random algorithm is used to assign time window for each block generator depending on its public key and the public key of the previous block generator hashed together.* *Stake must be stable for ~24 hours to be counted towards block generation to prevent manipulation.* *For security considerations see https://medium.com/@lyaffe/proof-of-stake-articles-cc6fbb346184*

Platform Ledger Management	
Model	*Balance*
Extra	*Child chains, Messages, Properties, Controls, Voting, Tokens, Marketplace, Cloud Data*
Description	*Rich state is maintained by the blockchain for each account including the main token balance, child chain token balances, auxiliary token balances, poll and voting state, account properties, on chain messages, marketplace balances, and more.*

Section 5 Resources

Node Management	
Node Role	*Full validating nodes and archival nodes*

Joining	*Setting up a new node does not require any permission. When started, a node will attempt to connect to a list of bootstrap nodes provided with the product installation. Once connected to another node it will query the remote node for additional nodes until reaching a configured number of open connections to remote nodes. A node operator can modify the list of bootstrap nodes before starting their node and connect/disconnect/blacklist nodes during runtime.*
Leaving	*Nodes may discontinue participation on the network at any time*
Role changing	*Nodes can independently change roles at any time*
Description	---

Platform Data Storage Mechanism	
Mass storage mitigation[1]	*Fee-based data storage pricing mitigates against mass on-chain data storage.*
Decentralized Data Storage Support	*Platform specific*
Data Privacy Solution	*Built-in encrypted messages based on AES* *On chain 'shuffling' to improve privacy is available on mainnet* *On-going Zero Knowledge Proof (ZKP) and Homomorphic Signatures research* *See https://ardordocs.jelurida.com/Arbitrary_messages and https://ardordocs.jelurida.com/Coin_Shuffling and* https://www.jelurida.com/standby-shuffling-explained
Tamper Proof (tamper cost)	*Immutable ledger. Storing encrypted data is supported.* *51% of the stake is needed in order to censor data. Transparent Forging provides higher protection. Since the identity of the next block generator can be determined with high probability, parties can send transactions directly to the block generator node or blacklist this node if it does not follow the consensus rules.*
Description	*Every transaction can have a plain text or encrypted attachment that can be used to store data, and also a dedicated "cloud data" transaction type exists. The network provides a mechanism to automatically request and restore pruned data from archival nodes that have stored it.*

Platform Network Management

[1] On chain storage cost much, solution/mechanism to resolve the problem of large cost of mass storage from node perspective. E.g., data maintenance, data storage and data cleaning.

Node Scalability	*No enforced limit*
Network Structure	*Distributed; Flexible*
Network Discovery Protocol	*Starting from a list of bootstrap nodes. Adding nodes based on nodes known to remote nodes.*
Byzantine Node Accepted?	*Yes*
P2P?	*Yes*
Data Exchange Protocol	*Gossip*
Description	*Data exchange between nodes relies on efficient peer to peer native socket communication network.*

Section 6 Utils

Platform Messaging Mechanism	
Protocol Type	*Proprietary*
Description	*Plain text messages and encrypted messages are supported. Messages can be prunable i.e. removed from the blockchain only leaving its hash, message file attachments are supported. Every transaction type can have a message attachment or encrypted "message to self" / memo attachment.*

Platform Crypto Libraries	
Secure Network Connection Type	*SSL; TLS; …*
Cipher Suites	*EC-KCDSA Curve25519*
Description	*Key exchange is based on the Curve25519 algorithm, which generates a shared secret key using a fast, efficient, high-security elliptic-curve Diffie-Hellman function. The algorithm was first demonstrated by Daniel J. Bernstein in 2006. Java-based implementations were reviewed by cryptography expert back in March, 2014.* *Message signing in Nxt is implemented using the Elliptic-Curve Korean Certificate-based Digital Signature Algorithm (EC-KCDSA), specified as part of IEEE P1363a by the KCDSA Task Force team in 1998.* *Both algorithms were chosen for their balance of speed and security for a key size of only 32 bytes.*

Section 7 Operation & Maintenance

Platform system management – Node

Log	*Yes*
Monitoring	*Yes*
Description	*Node operation can be controlled by the local operator or monitored by a public service like https://ardor.peerexplorer.com/ , nodes use extensive logging facility and advanced monitoring APIs.* *https://ardordocs.jelurida.com/API*

Platform system management – Chain Network	
Permission Control	*No permission is required*
Auditing	*Extensive audit is supported*
Supervisory Support	*Node automatic update is not supported as this would create a form of centralization. Node upgrade by itself is very simple and can be performed in batch by the node operator. Docker template is available.*
Description	

Section 8 External Resource Management

Platform External Resource Management	
Interoperation solution	*Transactional child chains are operational on mainnet since 1 January 2018.* *Unique transaction vouchers architecture enables secure offline communication of transaction data.* *Stateless Lightweight Contracts, Ardor's version of smart contracts, can be deployed on a child chain and used as clients to any external system which expose Java, Web Service, Restful or Http based APIs.* *Unlike other blockchain contract designs, the Ardor lightweight contracts are designed to interface and inter-operate with other external systems. Lightweight contracts are also designed to simplify contract life cycle management.*
Description	*Ardor is the first platform to release a fully operational multi-chain architecture to production since January 2018. Ardor's multi-chain architecture improves scalability, reduces blockchain bloat and provides several operational advantages like sponsored transaction fees.* *The new lightweight smart contract framework also offers a stateless solution for app creation on the blockchain.*

Section 9 Extensions

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Child chains*
Extension type[2]	*Internal*
Extension mode[3]	*Horizontal*
Solution	*[internal] child-chain*
Serve domain	*Horizontal: sharding*
Description	*Transactional child chains receive their security from the POS consensus on the decentralized Ardor "parent chain." Each child chain is connected to the Ardor parent chain by "bundlers." Bundlers operate as mini-exchanges – they collect fees on child chains in the native child chain token, and pay ARDR to validate the transactions on the Ardor parent chain. Anyone with 1000 ARDR or more can set up a bundler. Bundlers can be customized so they only package transactions originating from specific accounts or involving specific digital assets or monetary supplies. This design means child chain owners can sponsor their end user fees. Additionally, the Ardor network is never monopolized by an individual application since each block on the Ardor parent chain has a limit to the number of transactions it will bundle per child chain. Additionally, all child chains are compatible with one another.*

Platform Extensions - optional	
[the following list can be duplicated for multiple extensions]	
Name	*Stateless lightweight smart contracts*
Extension type[4]	*Internal*

[2] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

[3] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).

[4] Standing from DLT system instance perspective, any extension inside the instance is marked as "internal", while any extension outside the instance is marked as "external"

Extension mode[5]	*Vertical*
Solution	*[external] secure oracles that interface with off-chain databases*
Serve domain	*Vertical: storage*
Description	*Stateless Lightweight Contracts, Ardor's version of smart contracts, can be deployed on a child chain and used as clients to any external system which expose Java, Web Service, Restful or Http based APIs.* *Unlike other blockchain contract designs, the Ardor lightweight contracts are designed to interface and inter-operate with other external systems. Lightweight contracts are also designed to simplify contract life cycle management.*

[5] All extension instances are equal (with similar capability and functional features), targeting for the scalability of DLT instance, marked as "horizontal"; extensions with different functional features, targeting to enforce the capability of DLT instance, marked as vertical. Extension type and mode pair(s) is/are used to describe the extension as to the whole DLT system. E.g., sharding (internal – horizontal), lightening – BTC (external – vertical), Corda Contract (internal – vertical).