====== Getting started on ROKSNET ====== - Install **Security Server** and setup **Trust Services**: - - * **AUTH** Certificate for Security Server - - * **SIGN** Certificate for each Subsystem behind it - Connect Security Server with your information system (Adapter development (Java, .NET, PHP etc.)) - Configure data **access settings** (in Security Server) - Exchange data with your business partners **Tips:** * You can use ready **RoksNet Portal** software for end-users (employees, customers, citizens etc.) * Unordered List ItemUse RoksNet Portal for own services **testing purposes** {{:public:docs:roksnet:simple.png?900|}} ==== Information system owner view of architecture ==== RoksNet utilizes the X-Road components for secure data transfer. The illustration below shows the main components and interfaces of the RoksNet. {{ :public:x-road_architecture.png?direct&800 |}} ===== Important (to get a unique Member/User Code for RoksNet) ====== Before setting up your Security Server, please provide the following information to us: * **Entity name** * **Status (Goverment, Private Company, Non-Profit Organization)** * **Organization unique registry code (if exists)** * **Country** and official Address * Name of the person, who represents the organization in RoksNet * Representative identifier/unique personal ID code (if exists) The fields in **bold** text are required. Please send the following information to [[support@roksnet.com]] and we'll send you a 8-digit Member/User Code, that will be needed later to RoksNet setup process. ===== DOWNLOAD ===== ==== STEP 1 (required) - Setting up a Security Server and Trust Services ==== === System requirements === ^ Requirements ^ Explanation ^ | Ubuntu 14.04 LTS x86-64 | Operating system | | 2 GB RAM, 10 GB disk space | Minimum system requirements | | http://apt.roksnet.com/xroad6-debs | RoksNet package repository | | http://apt.roksnet.com/xroad6-debs/roksnet_repo.gpg | The repository key | | TCP 5500 | Port inbound & outbound for message exchange between security servers | | TCP 5577 | Port inbound & outbound for querying OCSP responses between security servers | | TCP 4001 | Port outbound for communication with the roksnet registry | | TCP 80 | Port outbound for downloading global configuration | | TCP 443 | Port outbound for OCSP and TSA services | | TCP 4000 | Port inbound for access to user interface (local network) | | TCP 80, 443 | Ports inbound as information system access points (local network or external via https) | REQUIREMENTS FOR THE SECURITY SERVER Minimum recommended hardware parameters: * the server’s hardware (motherboard, CPU, network interface cards, storage system) must be supported by Ubuntu 14.04 in general; * a 64-bit dual-core Intel, AMD or compatible CPU; AES instruction set support is highly recommended; * 2 GB RAM; * a 100 Mbps network interface card; * if necessary, interfaces for the use of hardware tokens. Requirements to software and settings: * an installed and configured Ubuntu 14.04 LTS x86-64 operating system (VMs are supported as long as they support Ubuntu 14.04 LTS); * a static ip address should be configured; * The enabling of auxiliary services which are necessary for the functioning and management of the operating system (such as DNS, NTP, and SSH) stay outside the scope of this guide. === Installing the Security Server === PREPARING THE OS 1. Add system user whom all roles in the user interface are granted to: <code>sudo adduser username</code> 2. Set operating system locale. Add the following line to /etc/environment: <code>LC_ALL=en_US.UTF-8</code> INSTALLATION 3. Add the address of the RoksNet package repository and the nginx repository to to /etc/apt/sources.list.d/roksnet.list: <code>deb [arch=amd64] http://apt.roksnet.com/xroad6-debs trusty main deb http://ppa.launchpad.net/nginx/stable/ubuntu trusty main deb http://ppa.launchpad.net/openjdk-r/ppa/ubuntu trusty main</code> 4. Add RoksNet's repository's signing key to the list of trusted keys: <code>curl http://apt.roksnet.com/xroad6-debs/roksnet_repo.gpg | sudo apt-key add -</code> <code>sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 00A6F0A3C300EE8C</code> <code>sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EB9B1D8886F44E2A</code> 5. Install the security server software: <code>sudo apt-get update</code> <code>sudo apt-get install xroad-securityserver</code> 6. During the installation you will be asked to specify the username (added at step 1.) that will be granted the rights to perform all activities in the user interface. 7. The other questions can be answered with their default values as they are detected from the OS. POST INSTALLATION CHECKS 8. Check if all the processes started. The following services should be running.

<code>sudo initctl list | grep "^xroad-" xroad-jetty start/running, process 19796 xroad-confclient start/running, process 19563 xroad-signer start/running, process 19393 xroad-proxy start/running, process 19580 </code> 9. Ensure that the Security Server user interface at https://SECURITYSERVER:4000/ can be opened in a web browser. To log in, use the account name chosen during the installation. While the user interface is still starting up, the web browser may display the “502 Bad Gateway” error. ==== Configuring the Security Server ==== 1. To perform the initial configuration, open the address https://SECURITYSERVER:4000/ in a web browser. To log in, use the account name chosen during the installation. 2. The system will ask for a global configuration anchor file. RoksNet's development environment's anchor can be downloaded from http://apt.roksnet.com/xroad6-debs/configuration_anchor.xml. The hash value for verification is <code>Hash (SHA-224): 71:3B:88:8C:6B:59:73:29:57:D2:06:8D:BE:A0:FF:F4:E4:E8:E1:D9:3A:A6:8A:75:E3:02:E7:FB</code> {{ :public:docs:roksnet:anchor_upload.png?direct&700 |}} 3. If the configuration is successfully downloaded, the system asks for the following information: * The security server owner’s member class (COM for private sector business, GOV for governmental organization and NGO for non-profit organization) * The security server owner’s Member/User code (this should be your organization's registry code) * **NB! Please enter an 8-digit Member/User code you got from RoksNet support. If you do not have one, please contact [[support@roksnet.com]] to get your code. More information at section [[public:start#important|important]] ** * Security server code (free form, security server code should be unique per member). * Software token’s PIN (free form - the PIN will be used to protect the keys stored in the software token. The PIN must be stored in a secure place, because it will be no longer possible to use or recover the private keys in the token once the PIN has been lost) Example (values in the picture are for demostration purposes): {{ :public:docs:roksnet:initialization.png?direct&700 |}} 4. The system will prompt a warning. This is fine, as we'll register the member status later. {{ :public:docs:roksnet:warning.png?direct&400 |}} 5. Enter softtoken PIN in the Keys and Certificates view (chosen in step 3.) {{ :public:docs:roksnet:softtoken.png?direct&700 |}} 6. Add a TSA in System Parameters view {{ :public:docs:roksnet:tsa.png?direct&700 |}} 7. Generate keys and certificate requests in the Keys and Certificates view Security Servers use 2 types of certificates * AUTH certificates for authentication between security servers when initiating a secure TLS channel. AUTH certificates are used 1 per security server. * SIGN certificates for e-Stamps. SIGN certificates are used 1 per Member/User (i.e. organization). Generate 2 keys by selecting the SoftToken-0 and press GENERATE KEY. Give the key a label and press OK. Your keys should now look similar to this: {{ :public:docs:roksnet:keys.png?direct&700 |}} Next, generate a CSR for a SIGN certificate by choosing a key and selecting GENERATE CSR. Make sure the Usage: SIGN is selected. If the Common Name (CN) field is empty, please use the Member/User Name in that field. {{ :public:docs:roksnet:sign.png?direct&700 |}} Then generate a CSR for an AUTH certificate. Make sure the Usage: AUTH is selected. If the Common Name (CN) field is empty, please use the Member/User Name in that field. {{ :public:docs:roksnet:auth.png?direct&700 |}} Send the downloaded CSR-s to us at [[support@roksnet.com]] with the following details: * Member/User name * Member/User code * Member/User class * Security server code We will use the CSR-s to issue you certificates. Once you have received the certificates you should be able to import them in the "Keys and Certificates" view. After importing them, select the AUTH certificate and press "**ACTIVATE**" and "**REGISTER**". Your keys and certificates should now look like this: {{ :public:docs:roksnet:keys2.png?direct&700 |}} Once we have accepted the registration request in the RoksNet User Registry, the OCSP response and status of the AUTH certificate will change to "good, registered". The next step would be registering a Subsystem. Select "ADD CLIENT" in the Security Server Clients view. {{ :public:docs:roksnet:subsystem.png?direct&300 |}} Select "CONFIRM" in the next prompt. {{ :public:docs:roksnet:confirm.png?direct&300 |}} Once we have accepted the registration request in the RoksNet User Registry, you'll be ready to Consume or Provide User Content Services on RoksNet's Development Environment. ==== STEP 2 (optional) - Setting up RoksNet Portal ==== RoksNet

Portal is a universal client application that's easy to set up and makes it easy to consume Content Services without the need to develop one's own information system. It's also a good developers tool, as it makes testing your services easy. === System requirements === ^ Requirements ^ Explanation ^ | Ubuntu 14.04 64-bit, 4 GB RAM | Recommended system requirements | | http://apt.roksnet.com/roksnet-portal | RoksNet Portal repository | | TCP 80, 443 outbound | For communicating with the security server (HTTPS if over public internet | | TCP 443 inbound | For access to the web interface | Requirements to software and settings: * an installed and configured Ubuntu 14.04 LTS x86-64 operating system (VMs are supported as long as they support Ubuntu 14.04 LTS) **NB! Needs a separate Ubuntu 14.04 LTS server - will not work if installed alongside the Security Server**; * a static ip address should be configured. * the enabling of auxiliary services which are necessary for the functioning and management of the operating system (such as DNS, NTP, and SSH) stay outside the scope of this guide. === Installing RoksNet Portal === 1. All activities during installation are performed as root user, so <code>sudo -i</code> 2. Configure RoksNet Portal and JDK-8 repositories. To do that, add the following lines to /etc/apt/sources.list.d/roksnet-portal.list <code>deb [arch=amd64] http://apt.roksnet.com/roksnet-portal trusty main deb http://ppa.launchpad.net/openjdk-r/ppa/ubuntu trusty main </code> 3. Download signing key for JDK-8 <code>apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EB9B1D8886F44E2A</code> 4. Update package lists <code>apt-get update</code> Ignore the following error for now: <code>W: GPG error: http://www.aktors.ee trusty Release: The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 671B319BE775D097</code> 5. Install RoksNet Portal keyring Install roksnet-keyring package: <code>apt-get install roksnet-keyring</code> When asked, choose "y" as an answer: <code>Install these packages without verification [y/N]?</code> Update the package lists again: <code>apt-get update</code> 6. Install postgreSQL <code>apt-get install postgresql-9.3</code> 7. Edit the file /etc/postgresql/9.3/main/pg_hba.conf so that postgres user's authenticaton method would be "trust". It should look like this: <code># Database administrative login by Unix domain socket local all postgres trust </code> 8. Restart PostgreSQL <code>service postgresql restart</code> 9. Install RoksNet Portal database package <code>apt-get install roksnet-postgresql</code> For testing purposes questions prompted during installation can be answered with default values. 10. Install Java JDK 8 <code>apt-get install openjdk-8-jdk</code> 11. Add JAVA_HOME variable <code>export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64</code> 12. Install RoksNet Portal application <code>apt-get install roksnet-application</code> Quite a few questions will be asked during this installation. * Please provide database host IP to be used - default answer * Please provide database port to be used - default answer * Please provide database name to be used - default answer * Please provide username to be communicating with database - default answer * Please enter username password - enter password chosen at step 9 * Please provide SMTP host address - answer default, won't be using for testing purposes * Please provide server email address - answer default * Do you want to add new administrator account - answer "y" * Do you want to enable HTTPS connection between RoksNet Portal application and security server? - answer "n" 13. Edit the file /var/lib/tomcat7/webapps/ROOT/WEB-INF/classes/config.cfg with the following content: You can change the country code: <code> #If no suitable countries are defined, then uses system default locale country countries = [ISO code of your country, e.g. "GB"] </code> 14. Edit /etc/apache2/sites-available/ssl.conf with your IP aadress to gain access to the admin interface. It should look like this: <code> <Location "/*/admin/*"> Order deny,allow Deny from all Allow from 127.0.0.1 192.168.1.90 </Location> </code> 15. Restart apache2. <code>service apache2 restart</code> === Configuring RoksNet Portal from the web interface === 1. You should now be able to log into RoksNet Portals's admin interface at https://server_ip/admin. Use the administrator account you created during installation to log in. 2. When logged in, under Portal management, select "Add new" 3. Fill out the forms with the following information: * Portal name (EN) - free form * Portal short name - system name, free form * Portal type - Organizations portal * Organization name (EN) - Name of your organization * Organization code (RoksNet User Code) - User/Member Code (must match

User/Member Code in Security Server) * Data protocol version - 4.0 * RoksNet instance - roksnet-dev * RoksNet member class - Your Member/User class (must match Member/User class in Security Server) * RoksNet subsystem code - Your Subsystem Code (must match Subsystem Code in Security Server) * Security host - http://security_server_ip * Services sending address - http://security_server_ip * Debug mode - off * Send audit log to security server - unchecked * Use topics - unchecked {{ :public:docs:roksnet:add_portal.png?direct&700 |}} After filling out the forms, hit "Save portal configuration". 4. Select "Add new manager". 5. Fill out the forms and hit "Add new person as manager". For password authentication, check the "change password" checkbox to see the new password fields. {{ :public:docs:roksnet:manager.png?direct&400 |}} 6. Choose "Exit" at upper right corner. 7. Log in with the user you just created at https://server_ip/ The username must be enter as [countrycode+serialnumber]. {{ :public:docs:roksnet:signin.png?direct&400 |}} 8. Select "All producers" -> "Refresh producers", check the following boxes and hit "Save active producers". {{ :public:docs:roksnet:producers.png?direct&700 |}} 9. Select a producer, hit Refresh services -> From security server (allowed). This will send a metaquery from RoksNet Portal to your Security Server, which will proxy it the the Producer's Security Server. As a response, the Producer's Security Server will send a list of allowed methods for your Subsystem. 10. Check all Services and hit Refresh XForms descriptions of selected Services -> Generate from Security Server. This will generate XForms based on the services WSDL-s. {{ :public:docs:roksnet:services.png?direct&700 |}} 11. Hit the play button on a Service to test it. You're now able to consume Content Services via RoksNet Portal :) 12. The following **demo Content Services** are available in RoksNet Development Environment: *** Population Registry** *** Passport database** *** Vehicle database** *** Company database** *** Prescription database** NB! In most of demo Content Services you can use % symbol as input parameter and you will receive a list of persons or companies. Take some person or company code from the list and use it in another demo queries. The successful result shows that you received a demo data from another RoksNet User. ==== STEP 3 (optional) - User Content Services development and own application integration with RoksNet ==== === Developer documentation === RoksNet User Content Services are based on SOAP. Information about X-road specific requirements are in the following documents: * Message Protocol v4.0 - http://x-road.eu/docs/x-road_message_protocol_v4.0.pdf * Service Metadata Protocol - http://x-road.eu/docs/x-road_service_metadata_protocol.pdf === Python example === Source code of example: {{ :public:docs:roksnet:pyxadapter_1.3.tgz |}} The demo application is written in Python. Installation instructions are in included in the package (see INSTALL.txt) The example SOAP server accepts input messages in both X-road protocol 3.1 and 4.0 and formats response message according to the same protocol which was used by the client. The application contains source code of Population Registry, Passport database, Vehicle database, Company database and Prescription database demo services which are available in RoksNet Development Environment. To create a new SOAP server for Content Service provider: * create SOAP server instance for SOAP server (example: pyxadapter/companydb_server/__init__.py) * include service registration function and define routing for new SOAP server in pyxadapter/__init__.py To create a new service in the SOAP server: * create service implementation function serve() (example: pyxadapter/companydb_server/simplequery.py) * import and registrate service implementation module (example: pyxadapter/companydb_server/__init__.py) * create service description WSDL (example: pyxadapter/static/companydb_v4.wsdl) * publish service in Security Server and grant access to clients To create SOAP client: * create X-road client class (corresponding to a service provider) with methods corresponding to services and start calling those methods (example: pyxadapter/client/companydb.py) === Java adapter example === GitHub link: [[https://github.com/petkivim/x-road-adapter-example]] All of the related documentation is on the GitHub page === C# example with X-tee.NET === Tool and examples can be downloaded from [[http://xtee.codeplex.com/]] __How to use X-tee.NET tool__ - Get WSDL (if you plan to consume existing services) or write WSDL (if you plan to provide services). - Use generator to generate code based on WSDL. - Use generated code in your project to implement SOAP client or server. __Using

generator__ Execute the generator program Xtee.Gen.exe and fill the form with necessary data. User interface is in Estonian only. The fields in the form have following meaning: * DTO-e väljund kataloog – output directory for generated DTO code * Serialiseerijate väljund kataloog – output directory for generated serializers (you may use the same directory as for DTO) * Vali wsdli asukoht – path to WSDL file to read * Adapter name – name of the class which will be generated * XroadObjectType, XroadInstance, MemberClass, MemberCode, SubsystemCode – attributes of the Roksnet User who provides Content Services * Näita detailsemat logi – show detailed log * Genereeri - generate Fill the gaps and press button „Genereeri“. If generation is successful, you will get generated code in the output directories. Based on this code, you can create and compile a new class library for using in your project. __Implementation of SOAP client__ Add generated class library, Xtee.Code.dll and log4net.dll into your application. Write client code to consume the services: <code> using Calculator.Xroad.Eu; using Xtee.XteeClient.Calc; namespace Client { class Program { static void Main(string[] args) { var client = new CalcAdapter(); // ID of authenticated end user client.XteeCommand.Configuration.UserId = "EE30101010007"; // call the service decimal result = client.Add(new Variables(3, 4)).Result.Value; Console.WriteLine(result); } } } </code> This code creates a SOAP client instance using the class name which was entered into the generator. Each call must include actual user ID who has been authenticated by your application. Finally, we will call the service – in this example, service name is Add and it’s parameters are numbers 3 and 4. Add <xtee.configuration> element to the configuration file of you project: <code> <configSections> <section name="xtee.configuration" type="Xtee.Core.Client.Config.ClientConfigurationSection, Xtee.Core"/> </configSections> <xtee.configuration proxyURL="http://my_security_server/cgi-bin/consumer_proxy" xRoadInstance="roksnet-dev" memberClass="COM" memberCode="12998179" subSystemCode="roksnet-consumer" objectType="SERVICE"> <xteeTypeAssemblies> <clear/> <add assemblyName="Name_of_generated_DTO_DLL"/> <add assemblyName="Name_of_generated_serializers_DLL"/> </xteeTypeAssemblies> </xtee.configuration> </code> Replace attributes of <xtee.configuration> with your own data (according to the certificate issued to your security server). Possible attributes are: * proxyURL - SOAP server URL (typically in your security server unless you want to test your local SOAP server) * storeMessages - (true/false) whether to save message log (default: false) * storagePath - path to save messages, may be relative to app or absolute * timeout - timeout of service call in milliseconds (default: 100000 (1 minute 40 seconds)) * objectType - attribute of the client element of message * xRoadInstance - instance which you have joined * memberClass - value from your certificate * memberCode - value from your certificate * subSystemCode - value from your certificate * userId - default value for user ID * issue - default value for issue * message-path-format - message log filename format * temporaryStoragePath - optional path for saving attachments (if not provided, storagePath will be used). If value is :inmemory: then attachments will be handled in memory only Attribute values may be set within the client code as well. __Implementation of SOAP server__ Add generated class library, Xtee.Code.dll and log4net.dll into your application. To provide a service, create service class, inheriting it from the generated interface of the service, and implement ProcessRequest method: <code> namespace Server.Services { public class ServiceMultiply : MultiplyServiceHandler { public MultiplyResponse ProcessRequest(MultiplyRequest input, MultiplyInputHeader header) { return new MultiplyResponse(input.Request, new Result(input.Request.VarX * input.Request.VarY)); } } } </code> The configuration file of your application must include <adapter.configuration> element: <code> <configSections> <section name="adapter.configuration" type="Xtee.Core.Adapter.Service.Config.AdapterServiceConfigurationSection, Xtee.Core"/> </configSections> <adapter.configuration storagePath="stored_messages" storeMessages="true"> <services> <clear/> <!—- Bind each service to the handler class which will serve it --> <add objectType="SERVICE" xRoadInstance="roksnet-dev" memberClass="COM" memberCode="12998179" subSystemCode="Calc" serviceCode="Add" version="v1"

servicehandler="Server.Services.ServiceAdd,Server"/> </services> <xteeTypeAssemblies> <clear/> <add assemblyName="Name_of_generated_DTO_DLL"/> <add assemblyName="Name_of_generated_serializers_DLL"/> </xteeTypeAssemblies> </adapter.configuration> </code> Replace attributes of <adapter.configuration> with your own data (according to the certificate issued to your security server). Possible attributes are: * commandInterceptor - full name of the class which implements ICommandInterceptor and comma-separated name of DLL where the class can be found * storeMessages - (true/false) whether to save message log (default: false) * storagePath - path to save messages, may be relative to app or absolute * message-path-format - message log filename format * temporaryStoragePath - optional path for saving attachments (if not provided, storagePath will be used). If value is :inmemory: then attachments will be handled in memory only