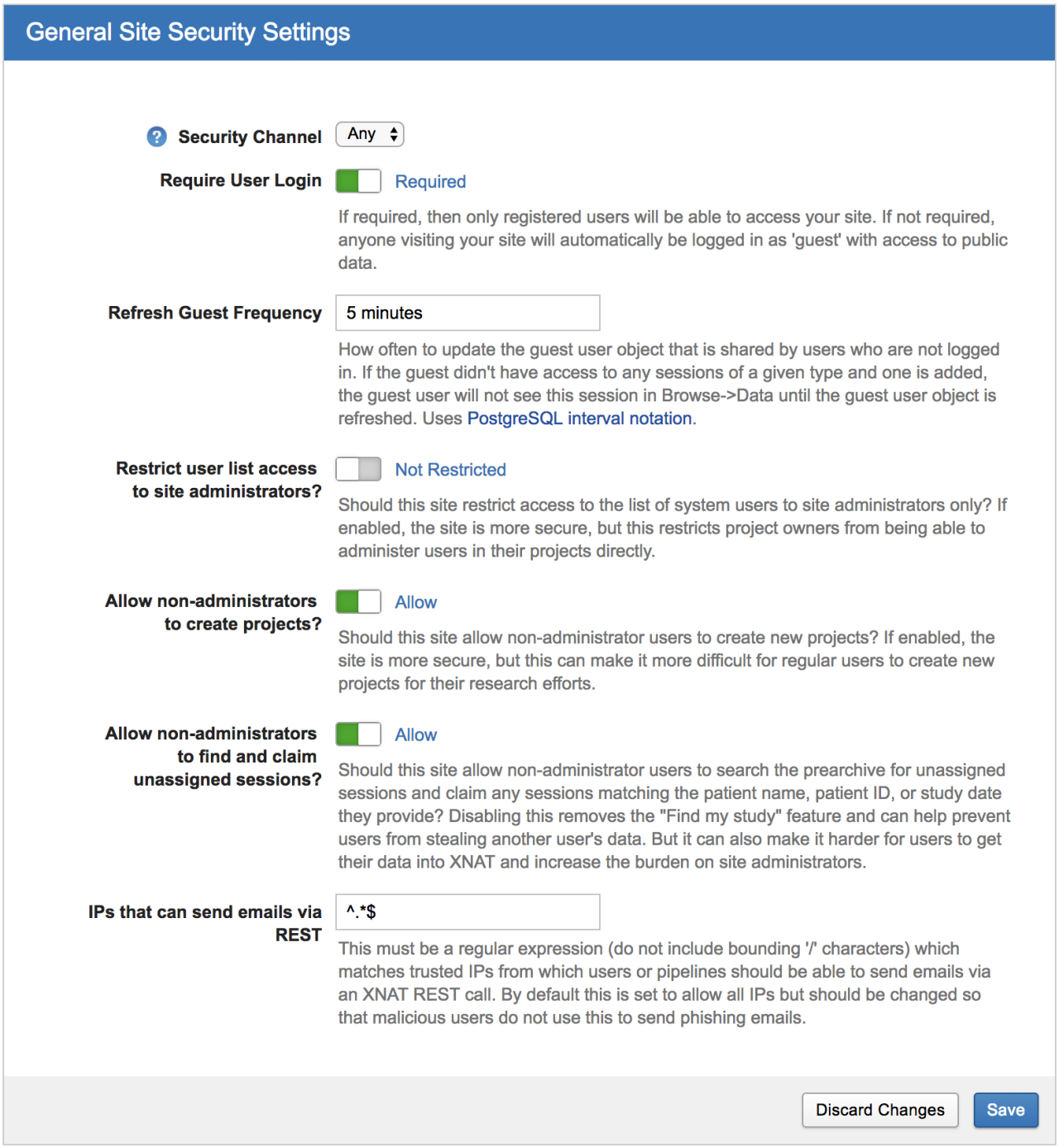
XNAT Security Settings in the Admin UI

Your security settings in XNAT will be dictated by the type of XNAT you plan to run. Are you setting up a publicly-accessible data repository like XNAT Central? Or a tightly-controlled data gathering and QC application for a multi-site study, like IntraDB for the Human Connectome Project? Your PI's business goals will need to be reflected in these security settings.

General Security Settings



Settings

Setting	Property Name	Description
Security Channel	`securityChannel`	This should be set to **http** if users should access your site with URLs starting in http. If you want users to access your site over secure HTTP and have a certificate, you should set this to **https**. Users would then access your site at URLs starting with https.
Require User Login	`requireLogin`	This determines whether users are required to be logged into an account in order to view the site's content. If set to Required (the toggle will be green), people will have to log in to an account (or register if they don't yet have one) in order to access the site. If set to Not Required, people that have not logged in will be treated as the guest user and will have access to all public data.
Refresh Guest Frequency	`refreshGuestFrequency`	How often to update the guest user object that is shared by users who are not logged in. If the guest didn't have access to any sessions of a given type and one is added, the guest user will not see this session in Browse->Data until the guest user object is refreshed. ⓘ This setting uses PostgreSQL interval notation (e.g. '1 day', '3 hours', '5 weeks', '1 year').
Restrict User List to Site Administrators	`restrictUserListAccessToAdmins`	This determines whether users other than those with site administrator permissions should be able to view the list of all users who have used the XNAT instance. If set to Restricted (the toggle will be green), only site administrators will be able to see the list of site users. This is more secure, but this can make it harder for users to add other users to their projects. If set to Not Restrcited, then any user of the site can see the list of all the users of the site.
Allow Non-administrators to Create Projects?	`uiAllowNonAdminProjectCreation`	This determined whether all users are able to create projects, or only site administrators. If set to Allow (the toggle will be green), then any user can create a project. If set to Do Not Allow, then only site administrators can create new projects. Do Not Allow is more appropriate for situations in which the XNAT instance is intended for distributing a limited set of data, while the Allow option makes more sense for XNAT instances which are intended to give large numbers of users a place to share their data.
Allow non-administrators to find and claim unassigned sessions?	`allowNonAdminsToClaimUnassignedSessions`	Should this site allow non-administrator users to search the prearchive for unassigned sessions and claim any sessions matching the patient name, patient ID, or study date they provide? Disabling this removes the "Find my study" feature and can help prevent users from stealing another user's data. But it can also make it harder for users to get their data into XNAT and increase the burden on site administrators.
IPs that can send emails via REST	`ipsThatCanSendEmailsThroughRest`	This must be a regular expression (do not include bounding '/' characters) which matches trusted IPs from which users or pipelines should be able to send emails via an XNAT REST call. By default this is set to allow all IPs but should be changed so that malicious users cannot use this to send phishing emails. ✓ Multiple IPs should be separated by pipe ("|") characters.

✓ Site-wide config properties can be accessed via REST at `/xapi/siteConfig/values/{property-name}`.

Site-wide config properties can be access programmatically in Velocity via `$siteConfig.{property-name}`.

User Logins and Session Controls

User Logins / Session Controls

Session Timeout
15 minutes
Interval for timing out user sessions. Uses PostgreSQL interval notation.

Alias Token Timeout
2 days
Interval for timing out alias tokens. Uses PostgreSQL interval notation.

Alias Token Timeout Schedule
0 0 * * * *
How often to check alias tokens for timeout (0 0 * * * * means it runs every hour). Uses basic Cron notation (lists and ranges aren't supported).

Session Timeout Message
```
Session timed out at TIMEOUT_TIME.
```
Alert message provided to users after a session timeout. TIMEOUT_TIME will be replaced by the timeout time.

Maximum Concurrent Sessions
1000
The maximum number of permitted sessions a user can have open simultaneously. You must restart Tomcat for changes to this to take effect.

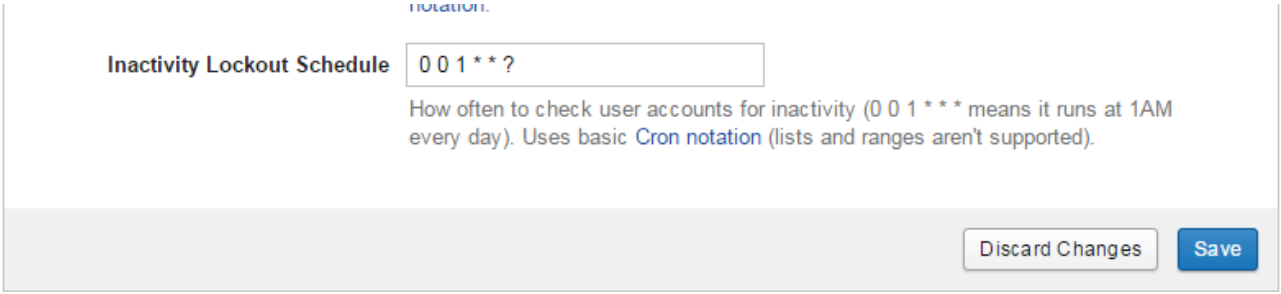
Login Failure Message
```
Your login attempt failed because the username and password
combination you provided was invalid or the site is
currently at the maximum number of user sessions. After %d
failed login attempts, your user account will be locked. If
you believe your account is currently locked, you can:<ul>
<li>Unlock it by resetting your password</li><li>Wait one
hour for it to unlock automatically</li></ul>
```
Text to show when a user fails to login

Maximum Failed Logins
5
Number of failed login attempts before accounts are temporarily locked. (-1 disables feature)

Failed Logins Lockout Duration
1 hour
Interval of time to lock user accounts that have exceeded the max_failed_logins count. Uses PostgreSQL interval notation.

Reset Failed Logins Schedule
0 0 * * * *
How often to check if the Failed Logins Lockout Duration time has expired so locked out users can be allowed to log in again (0 0 * * * * means it runs every hour). Uses basic Cron notation (lists and ranges aren't supported).

User Inactivity Lockout
1 year
Interval of inactivity before a user account is disabled. Uses PostgreSQL interval notation



User Session Settings

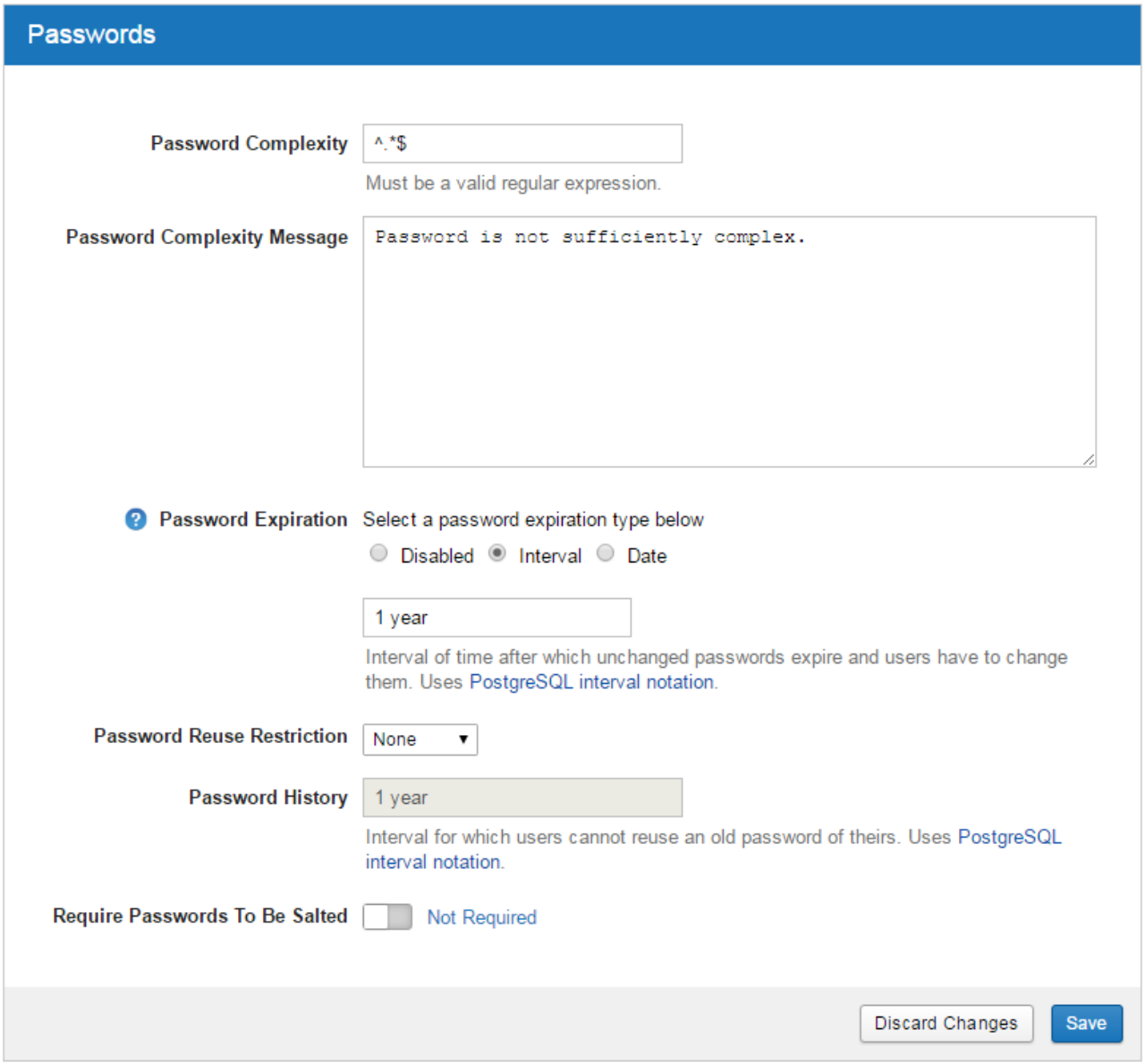
Setting	Property Name	Description
Session Timeout	`sessionTimeout`	This controls how quickly user sessions should time out. If this is set to "15 minutes", users who have not interacted with the site for 15 minutes will be automatically logged out. When users go to new pages in XNAT, their auto-logout timer in the upper right corner or the screen resets to whatever the session timeout is set to ("15 minutes" by default). Entering text into a text box is not sufficient to reset the timer, but submitting it is. Users can always click 'renew' in the upper right to reset this timer and when a user is getting close to being timed out, there will be a warning modal and they will have a chance to then renew their session. So users who want to spend a long time filling out a single form have ways to avoid timing out. When the session timeout is changed, it will take effect for all future users who log in (a Tomcat restart is not necessary for this to take effect). But users who are already logged in will still have the old auto-logout interval until they log out and log back in. ⓘ This setting uses PostgreSQL interval notation (e.g. '1 day', '3 hours', '5 weeks', '1 year').
Alias Token Timeout	`aliasTokenTimeout`	This controls how quickly alias tokens should time out. If this is set to "2 days", alias tokens that are more than 2 days old will no longer be valid. Long time intervals pose greater security risks, but short time intervals could pose problems if you are trying to do processing using an alias token and do not have enough time to complete it before the alias token times out. This takes effect immediately, and existing alias tokens will be invalidated if they were created more than the interval amount of time ago (e.g. setting this to '1 hour' will invalidate alias tokens created more than an hour ago). However, all alias tokens are not checked every second to see whether they have expired. You can control how frequently this is checked by changing the Alias Token Timeout Schedule preference. The length of time between checking for alias token expiration should be less than the length of time after which alias tokens expire. For example, if you set the alias token timeout to '2 hours', but set the alias token timeout schedule to run once a day, some alias tokens will remain valid for up to 26 hours (2 + 24), while others will be invalidated after two hours (if the daily check happens right after an alias token completes its second hour of existence). ⓘ This setting uses PostgreSQL interval notation (e.g. '1 day', '3 hours', '5 weeks', '1 year').
Alias Token Timeout Schedule	`aliasTokenTimeoutSchedule`	This controls how often alias tokens should be checked to see whether they have expired (invalidating those that have). This preference is explained someone in the previous Alias Token Timeout section, which is heavily related. Alias Token Timeout basically controls how old an alias token has to be before XNAT thinks it's ready to expire, and Alias Token Timeout Schedule controls how often XNAT should go through its alias tokens and invalidate all those that are ready to expire. Like with Alias Token Timeout, changing the Alias Token Timeout Schedule will take effect immediately.
Session Timeout Message	`sessionTimeoutMessage`	This controls what the message will be when a user's session times out. After being logged out, they will be redirected to the login page and this message will show up. If you want to refer to the exact time at which the user was logged out, simply include the string: TIMEOUT_TIME. Any instances of that string will be replaced by the actual timeout time (e.g. Wed Oct 26 21:21:05 UTC 2016). By default the session timeout message is set to "Session timed out at TIMEOUT_TIME." Changes to the session timeout message will take effect immediately, with the new message even displaying when your current session times out.
Maximum Concurrent Sessions	`concurrentMaxSessions`	This controls how many sessions a single user can have open at one time. By default this is set to 1000. If you want to run a large number of pipelines at the same time as a single user, you may need to increase this. Or if you want to prevent users from opening a lot of sessions, you might want to use a much smaller number. Any changes to this number will not take effect until you restart Tomcat. This means that changing this setting will not interfere with any existing sessions.

Login Failure Message	`uiLoginFailureMessage`	This is the message that will show up whenever a user tries to log in to the site and the login fails. This will usually be because the username/password combination was incorrect. This could also be because they had previously entered so many incorrect passwords that they are locked out of their account. If the site has multiple authentication providers, then it may also be the case that the user entered a valid username/password combination but selected the wrong provider. Another possibility is that the user trying to log in has already hit the maximum number of concurrent sessions they are allowed to have.
		
		If the login failure message contains **%d**, then the **%d** will be replaced with the value for the maximum failed logins preference. You are able to use HTML tags in your message, though we have not tested messages with very complicated HTML.
		
		By default, the login failure message is set to:
		
		`Your login attempt failed because the username and password combination you provided was invalid or the site is currently at the maximum number of user sessions. After %d failed login attempts, your user account will be locked. If you believe your account is currently locked, you can:`
		`<ul>`
		`  <li>Unlock it by resetting your password</li>`
		`  <li>Wait one hour for it to unlock automatically</li>`
		`</ul>`

Maximum Failed Logins	`maxFailedLogins`	This is the number of consecutive times that a user can fail to log in before they get locked out. By default this is set as **5**, meaning that if a user incorrectly enters their password 5 times, but then gets it right on their sixth try, they will not be logged in because they their account will have been locked after their fifth attempt. This count resets after every successful login. So a user that gets their password correct every fifth time would never be locked out. If you do not want users to ever be locked out for failed logins, you should set this to **-1**. This would make it less likely that valid users would be locked out of their accounts, but would make it easier for people who should not have access to the site to get in by guessing a large number of passwords (which can be done in an automated way). Changes to this setting take effect immediately. If you raise the number of maximum failed logins, users who had hit the old maximum number of failed logins, but who had fewer failed logins than the new maximum should be able to log in again.
Failed Logins Lockout Duration	`maxFailedLoginsLockoutDuration`	This is the time period users should be locked out if they hit the maximum number of failed logins. By default this is set to '1 hour'. Once this time period is over, there is a task that will reset their number of failed login attempts to 0. The frequency with which this task runs is controllable by changing the 'Reset Failed Logins Schedule'. By default, that is also set to an hour, which means that every hour, all users who have not tried to log in in the last hour will have their number of failed login attempts reset to 0. If the 'Reset Failed Logins Schedule' remains set to run every hour, and you change the 'Failed Logins Lockout Duration' to '1 day', then every hour, users who have not tried to log in in the last day will have their failed login attempts reset to 0 (and will be able to log in). ⓘ This setting uses PostgreSQL interval notation (e.g. '1 day', '3 hours', '5 weeks', '1 year').
Reset Failed Logins Schedule	`resetFailedLoginsSchedule`	This controls how frequently users whose lockout has expired are re-enabled. A task will run and reset the failed login counts of all users who have already been locked out for the Failed Logins Lockout Duration. This will only re-enable users that have been locked out due to failed logins, not those who have been locked out due to inactivity, or users who have explicitly been disabled by an admin. ⓘ This schedule should be expressed in Cron notation. By default it is set to "0 * * * *", which means that it will run every hour.
Reset failed login count on forgot password?	`canResetFailedLoginsWithForgotPassword`	Indicates whether a user should be able to click the link from a forgot password email to reset their count of failed logins. If enabled, a locked out user will be able to request a forgot password email and click the link from that email to change their password and unlock their account.
User Inactivity Lockout	`inactivityBeforeLockout`	This is the amount of time a user can be inactive before they get locked out of the site and have to re-enable their account. Users who have not logged in for this period of time will be locked out the next time the scheduled task that locks out inactive users runs (the frequency of this task running is controlled by the Inactivity Lockout Schedule). By default this is set to '1 year'. ⓘ This setting uses PostgreSQL interval notation (e.g. '1 day', '3 hours', '5 weeks', '1 year').

Inactivity Lockout Schedule	`inactivityBeforeLockoutSchedule`	This controls how frequently inactive users are locked out. A task will run and lock out users who have not logged in in the User Inactivity Lockout time period. For example, if the User Inactivity Lockout is set to '1 year' and Inactivity Lockout Schedule is set to "0 0 1 * * ?" (which are the defaults), then a task will run at 1AM every night and lockout all users that have not logged in in the last year. This means that once users have been inactive for a year, they will be locked out sometime in the next 24 hours. ⓘ This schedule should be expressed in Cron notation. By default it is set to "0 * * * * *", which means that it will run every hour.
Interactive Agent IDs	`interactiveAgentIds`	Regular-expression patterns that define user-agent header values that indicate whether a particular request comes from an interactive agent (e.g. a browser) or some other kind of tool, such as curl or other scriptable or automatable client. Separate multiple agent IDs with commas. The default setting is: `.*MSIE*., .*Mozilla*., .*AppleWebKit*., .*Opera*.`
Data Paths	`dataPaths`	Ant patterns that define URL patterns that represent data paths (e.g. REST API calls). XNAT uses these paths in conjunction with the interactive agent IDs above to determine the response to certain calls. For example, instead of simply returning a "404 Not Found" error when an inaccessible resource is indicated, a call to a data path by a browser may be redirected to the login page. Separate multiple data paths with commas. The default setting is: `/xapi/**, /data/**, /REST/**, /fs/**`

Passwords

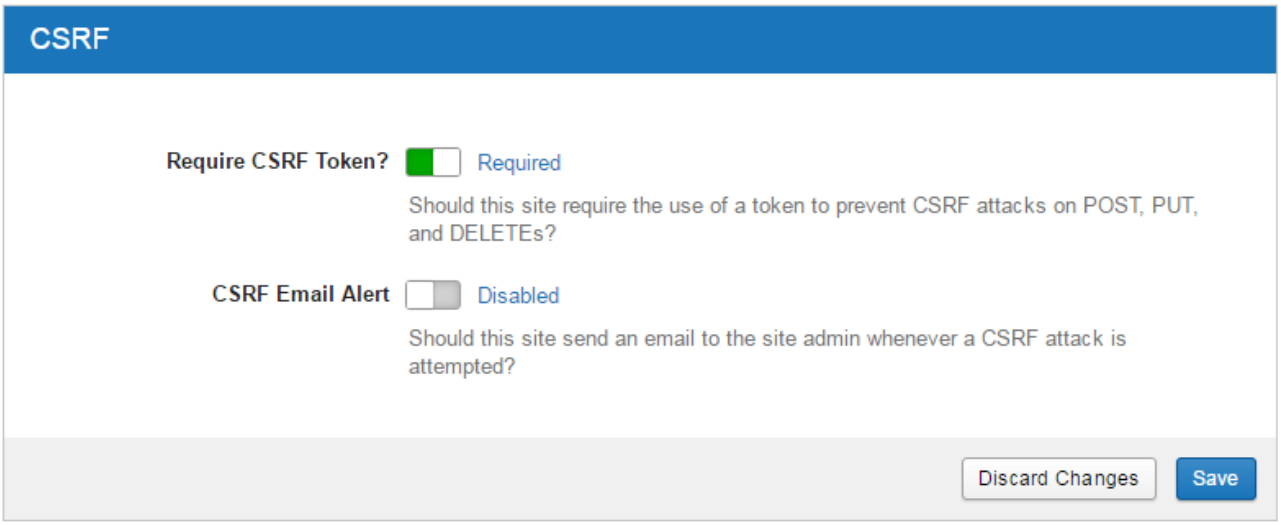


Password Settings

Setting	Property Name	Description
Password Complexity	`passwordComplexity`	This is a regular expression that controls what passwords will be considered sufficiently complex. When users register or change their password, they are required to choose a password that matches this regular expression. If their new password does not match the regular expression, they will be prompted to choose a new one. By default this regular expression is set to **`^.*$`**, which will match any password String. If you want to require that users choose more complicated passwords, you can change this to whatever Java regular expression you like. For example, this regex requires a password of at least 8 characters, with upper and lower case, a number, and a non-numeric symbol: `^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.[\W]).{8,}$`
Password Complexity Message	`passwordComplexityMessage`	This is the message that users will receive if they try to choose a new password that does not match the Password Complexity regular expression. By default this is set to 'Password is not sufficiently complex.', but you may want to change this to include a description of what is needed to satisfy the regular expression. For example, you may want this message to be 'Passwords must be at least 8 characters long and contain at least one digit'.

Password Expiration	`passwordExpiration`	This set of radio buttons controls password expiration. This setting can be set to: • **Disabled**: Passwords will never expire and users can continue to use the same passwords for as long as the site exists • **Interval**: Passwords expire after a specified interval of time • **Date:** Any passwords that were last changed before the specified date will be considered expired This setting exposes related settings, as defined below
Password Expiration (Interval)	`passwordExpirationInterval`	This setting is only used if Password Expiration is set to "Interval". It represents the interval of time after which unchanged passwords expire and users have to change them. By default this interval is set to '1 year', which means that users can continue using their password for a year before being required to change it. Having a short expiration interval can help protect against people who find someone's old password (e.g. if a user's password is included in requests which get written to a log file that a malicious user gets access to) by making it less likely that their old password is still valid. However, the risk of a short interval is that it can lead to people writing down their passwords or choosing new passwords that are nearly identical to their old passwords. ⓘ This setting uses PostgreSQL interval notation (e.g. '1 day', '3 hours', '5 weeks', '1 year').
Password Expiration (Date)	`passwordExpirationDate`	This setting is only used if **Password Expiration** is set to "Date". It represents the interval of time after which unchanged passwords expire and users have to change them. This can be useful when upgrading from an old version of XNAT to ensure that user passwords are stored with the latest security improvements. It can also be useful to expire by date if there is concern that malicious users might have been able to get access to passwords before that date (e.g. if before a certain date you had been writing user passwords to log files which were archived in an unsecure place).
Password Reuse Restriction	`passwordReuseRestriction`	This is what determines whether users are able to reuse old passwords. If set to 'None', then users will be able to reuse passwords they have used previously without any restrictions. If set to 'Historical', then users will be unable to change their password to a password that had been used within the time period specified in the Password History preference.
Password History	`passwordHistoryDuration`	This setting is only used if **Password History** is set to "Historical". This is the period of time for which users cannot reuse passwords. By default the Interval is set to '1 year', so users cannot change their password to any password they used in the last year. Once it has been a year since they used a given password, they can use it again. ⓘ This setting uses PostgreSQL interval notation (e.g. '1 day', '3 hours', '5 weeks', '1 year').
Require Passwords To Be Salted	`requireSaltedPasswords`	This controls whether users whose passwords are not currently salted in the database will need to change their password. Whenever users register or change their password, their passwords will be salted and then hashed before being stored in the database. However, when migrating users from earlier versions of XNAT, there may be some users who have not changed their password since XNAT started salting all passwords. If this setting is set to 'Required', these users will be required to change their password when they first log in. In addition, the default XNAT admin user, 'admin', does not have a salted password, so if this is set to 'Required', the admin will have to change their password (which they should be doing anyway). Having all user passwords be salted makes it harder for a user to discover what a user's password is, even if they have access to the users database table.

Cross-site Request Format (CSRF)

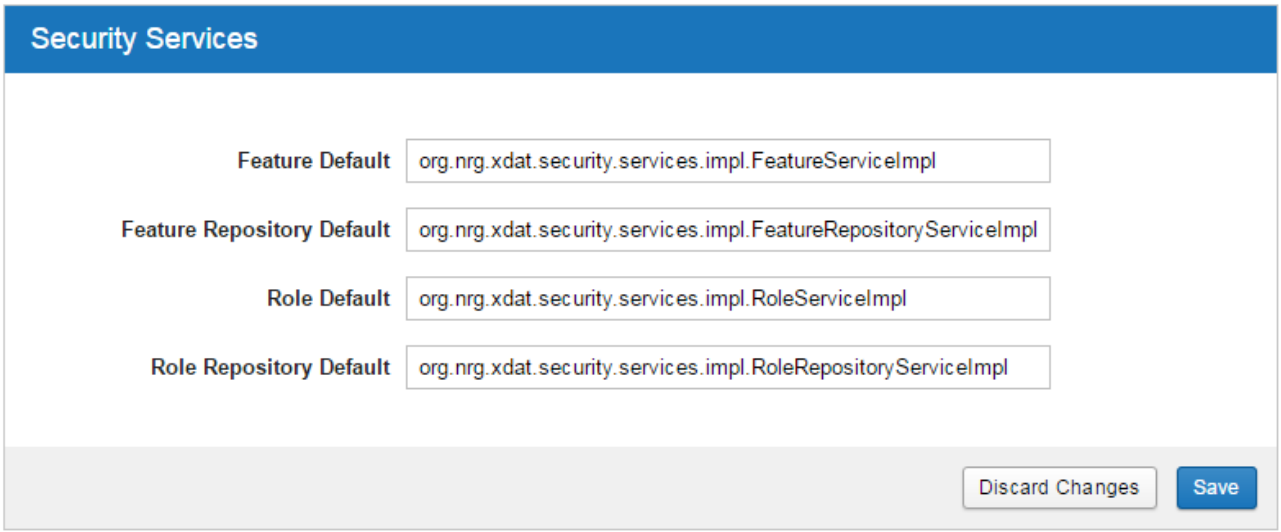


CSRF Settings

Setting	Property Name	Description
Require CSRF Token?	`enableCsrfToken`	This controls whether the site should require CSRF tokens. CSRF tokens are important for preventing Cross-Site Request Forgery attacks on your site. By default this is set to 'Required', which means that CSRF tokens will be used, and forged requests from malicious users will be ignored since they will not have the necessary CSRF token. If this is set to required, CSRF attacks on POST, PUT, and DELETE REST calls will be prevented. If set to 'Not Required', then your site will likely be vulnerable to Cross-Site Request Forgery attacks.
CSRF Email Alert	`csrfEmailAlert`	This controls whether the site admin should receive an email every time a possible CSRF attack is taking place. By default, this is set to 'Disabled' (meaning the site admin is not alerted of possible CSRF attacks), but should be enabled if the site admin is particularly concerned about Cross-Site Request Forgery attacks.

Security Services (Advanced)

⚠ These settings are related to functions that are not fully tested or supported in XNAT 1.7



Setting	Property Name	Description
Feature Default	`featureService`	This allows you to swap out which Java class is used for controlling access to features. Changes to this should take effect immediately. By default it is set to 'org.nrg.xdat.security.services.impl.FeatureServiceImpl'. You must make sure that whatever you change this to is a valid class that the webapp can access.
Feature Repository Default	`featureRepositoryService`	This allows you to swap out which Java class is used as the repository for features data. Changes to this should take effect immediately. By default it is set to 'org.nrg.xdat.security.services.impl.FeatureRepositoryServiceImpl'. You must make sure that whatever you change this to is a valid class that the webapp can access.
Role Default	`roleService`	This allows you to swap out which Java class is used for checking and changing user roles. Changes to this should take effect immediately. By default it is set to 'org.nrg.xdat.security.services.impl.RoleServiceImpl'. You must make sure that whatever you change this to is a valid class that the webapp can access.
Role Repository Default	`roleRepositoryService`	This allows you to swap out which Java class is used as the repository for storing user roles. Changes to this should take effect immediately. By default it is set to 'org.nrg.xdat.security.services.impl.RoleRepositoryServiceImpl'. You must make sure that whatever you change this to is a valid class that the webapp can access.