Configuration Tag	Configuration Name	New Config	Special Comments
JDBCPersistenceManager.DataSource	***Name***	[identity] data_source="jdbc/WSO2IdentityDB"	
JDBCPersistenceManager	***SkipDBSchemaCreation***	[data_source] skip_db_schema_creation=false	i
JDBCPersistenceManager.SessionDataPersist	***Enable***	[session_data.persistence] enable_persistance = true	
JDBCPersistenceManager.SessionDataPersist	***Temporary***	[session_data.persistence] persist_temporary_data = true	
JDBCPersistenceManager.SessionDataPersist	***PoolSize***	[session_data.persistence] persistance_pool_size = true	
JDBCPersistenceManager.SessionDataPersist.SessionDataCleanUp	***Enable***	[session_data.cleanup] enable_expired_data_cleanup = true	
JDBCPersistenceManager.SessionDataPersist.SessionDataCleanUp	***CleanUpTimeout***	[session_data.cleanup] expire_session_data_after = true	
JDBCPersistenceManager.SessionDataPersist.SessionDataCleanUp	***DeleteChunkSize***	[session_data.cleanup] clean_expired_session_data_in_chunks_of = true	
JDBCPersistenceManager.SessionDataPersist.OperationDataCleanUp	***Enable***	[session_data.cleanup] clean_logged_out_sessions_at_immediate_cycle= true	
JDBCPersistenceManager.SessionDataPersist.TempDataCleanup	***Enable***	[session_data.cleanup] enable_pre_session_data_cleanup= true	
JDBCPersistenceManager.SessionDataPersist.TempDataCleanup	***PoolSize***	[session_data.cleanup] pre_session_data_cleanup_thread_pool_size= true	
JDBCPersistenceManager.SessionDataPersist.TempDataCleanup	***CleanUpTimeout***	[session_data.cleanup] expire_pre_session_data_after= true	
TimeConfig	***SessionIdleTimeout***	[session.timeout] idle_session_timeout= true	
TimeConfig	***RememberMeTimeout***	[session.timeout] remember_me_session_timeout= true	
Security	***KeyStoresDir***	[key_mgt] keystore_dir= true	
	KeyManagerType	[key_mgt] key_manager_type= true	
	TrustManagerType	[key_mgt] trust_manager_type= true	
Identity	***IssuerPolicy***	[identity] issuer_policy= true	
	TokenValidationPolicy	[identity] token_validation_policy= true	
OAuth.TokenCleanup	***EnableTokenCleanup***	[oauth.token_cleanup] enable= true	
OAuth.TokenCleanup	***RetainOldAccessToken***	[oauth.token_cleanup] retain_access_tokens_for_auditing= true	
OAuth	***IdentityOAuthTokenGenerator***	[oauth.token_generation] access_token_type= "self_contained"	if access_token_type= "self_contained" then IdentityOAuthTokenGenerator = "org.wso2.carbon.identity.oauth2.token.JWTTokenIssuer" if access_token_type not defined default will be "org.wso2.carbon.identity.oauth2.token.OauthTokenIssuerImpl"

	AccessTokenValueGenerator	[oauth.token_generation]
		access_token_value_type= "uuid" #md5 #sha256
			if access_token_value_type= "uuid" then AccessTokenValueGenerator = "org.apache.oltu.oauth2.as.issuer.UUIDValueGenerator"
			if access_token_value_type= "md5" then AccessTokenValueGenerator = "org.apache.oltu.oauth2.as.issuer.MD5Generator"
			if access_token_value_type= "sha256" then AccessTokenValueGenerator = "org.wso2.carbon.identity.oauth.tokenvaluegenerator.SHA256Generator"
			if access_token_value_type not defined default will be "org.apache.oltu.oauth2.as.issuer.UUIDValueGenerator"

	UseSPTenantDomain	[authentication]
		sign_auth_response_with_tenant_of= "sp" #user
			if sign_auth_response_with_tenant_of= "sp" then UseSPTenantDomain = true
			if sign_auth_response_with_tenant_of= "user" then UseSPTenantDomain = false

	OAuth1RequestTokenUrl	[oauth.endpoints] oauth1_request_token_url= ""	
	OAuth1AuthorizeUrl	[oauth.endpoints] oauth1_authorize_url= ""	
	OAuth1AccessTokenUrl	[oauth.endpoints] oauth1_access_token_url= ""	
	OAuth2AuthzEPUrl	[oauth.endpoints] oauth2_authz_url= ""	
	OAuth2TokenEPUrl	[oauth.endpoints] oauth2_token_url= ""	
	OAuth2RevokeEPUrl	[oauth.endpoints] oauth2_revoke_url= ""	
	OAuth2IntrospectEPUrl	[oauth.endpoints] oauth2_introspect_url= ""	
	OAuth2UserInfoEPUrl	[oauth.endpoints] oauth2_user_info_url= ""	
	OIDCCheckSessionEPUrl	[oauth.endpoints] oidc_check_session_url= ""	
	OIDCLogoutEPUrl	[oauth.endpoints] oidc_logout_url= ""	
	OAuth2ConsentPage	[oauth.endpoints] oauth2_consent_page= ""	
	OAuth2ErrorPage	[oauth.endpoints] oauth2_error_page= ""	
	OIDCConsentPage	[oauth.endpoints] oidc_consent_page= ""	
	OIDCLogoutConsentPage	[oauth.endpoints] oidc_logout_consent_page= ""	
	OIDCLogoutPage	[oauth.endpoints] oidc_logout_page= ""	
	OIDCWebFingerEPUrl	[oauth.endpoints] oidc_web_finger_url= ""	
	OAuth2DCREPUrl	[oauth.endpoints] oauth2_dcr_url= ""	
	OAuth2JWKSPage	[oauth.endpoints] oauth2_jwks_url= ""	
	OIDCDiscoveryEPUrl	[oauth.endpoints] oidc_discovery_url= ""	
	UseEntityIdAsIssuerInOidcDiscovery	[oauth] use_entityid_as_issuer_in_oidc_discovery= ""	
	AuthorizationCodeDefaultValidityPeriod	[oauth.token_validation] authorization_code_validity= ""	
	AccessTokenDefaultValidityPeriod	[oauth.token_validation] app_access_token_validity= ""	
	UserAccessTokenDefaultValidityPeriod	[oauth.token_validation] user_access_token_validity= ""	
	RefreshTokenValidityPeriod	[oauth.token_validation] refresh_token_validity= ""	
	TimestampSkew	[oauth] timestamp_skew= ""	

	RenewRefreshTokenForRefreshGrant	[oauth.token_renewal] renew_refresh_token= ""	
	TokenPersistenceProcessor	[oauth] hash_tokens_and_secrets= true #false	`<!-- <module ref="addressing"/> -->`
	HashAlgorithm	[oauth] hash_token_algorithm= ""	
	EnableClientSecretHash	[oauth] hash_tokens_and_secrets= true #false	
	MapFederatedUsersToLocal	[oauth] map_federated_users_to_local= true #false	
	RedirectToRequestedRedirectUri	[oauth] oauth.redirect_to_idp_error_page_on_error = "true"	

OAuth.SupportedResponseTypes.SupportedResponseTypes	token	[oauth.reponse_type.token] # Advanced
		enable = true
		class = "org.wso2.carbon.identity.oauth2.authz.handlers.AccessTokenResponseTypeHandler"	

	code	[oauth.reponse_type.code] # Advanced
		enable = true
		class = "org.wso2.carbon.identity.oauth2.authz.handlers.CodeResponseTypeHandler"	

	id_token	[oauth.reponse_type.id_token] # Advanced
		enable = true
		class = "org.wso2.carbon.identity.oauth2.authz.handlers.IDTokenResponseTypeHandler"	

	id_token token	[oauth.reponse_type.id_token_token] # Advanced
		enable = true
		class = "org.wso2.carbon.identity.oauth2.authz.handlers.IDTokenTokenResponseTypeHandler"	

	custom response type	[[oauth.custom_response_type]] # Advanced
		name = "foo_response_type"
		class = "com.example.oauth.CustomResponseType"	

OAuth.SupportedGrantTypes.SupportedGrantType	authorization_code	[oauth.grant_type.authorization_code]
		enable = true
		grant_handler = "org.wso2.carbon.identity.oauth2.token.handlers.grant.AuthorizationCodeGrantHandler"	

	password	[oauth.grant_type.password]
		enable = true
		grant_handler = "org.wso2.carbon.identity.oauth2.token.handlers.grant.PasswordGrantHandler"	

	refresh_token	[oauth.grant_type.refresh_token]
		enable = true
		grant_handler = "org.wso2.carbon.identity.oauth2.token.handlers.grant.RefreshGrantHandler"	

	client_credentials	[oauth.grant_type.client_credentials]
		enable = true
		grant_handler = "org.wso2.carbon.identity.oauth2.token.handlers.grant.ClientCredentialsGrantHandler"
		allow_refresh_tokens = false
		allow_id_token = false	

	saml_bearer	[oauth.grant_type.saml_bearer]
		enable = true
		grant_handler = "org.wso2.carbon.identity.oauth2.token.handlers.grant.saml.SAML2BearerGrantHandler"	

	iwa:ntlm	[oauth.grant_type.iwa_ntlm]
		enable = true
		grant_handler = "org.wso2.carbon.identity.oauth2.token.handlers.grant.iwa.ntlm.NTLMAuthenticationGrantHandler"	

	jwt_bearer	[oauth.grant_type.jwt_bearer]
		enable = true
		grant_handler = "org.wso2.carbon.identity.oauth2.grant.jwt.JWTBearerGrantHandler"
		grant_validator = "org.wso2.carbon.identity.oauth2.grant.jwt.JWTGrantValidator"	

	uma_ticket.	[oauth.grant_type.uma_ticket]
		enable = true
		grant_handler = "org.wso2.carbon.identity.oauth2.token.handlers.grant.AuthorizationCodeGrantHandler"
		grant_validator = ""	

	custom grant type	[[oauth.custom_grant_type]]
		name = "name"
		grant_handler = "full qualified class name of grant handler"
		grant_validator = "validator"
		[oauth.custom_grant_type.properties]
		IdTokenAllowed = true	

OAuth	***OAuthCallbackHandlers***	[oauth.extensions] callback_handlers = ["org.wso2.carbon.identity.oauth.callback.DefaultCallbackHandler"]	

OAuth.TokenValidators	***bearer***	[oauth.token_validator.bearer]
		enable = true
		class = "org.wso2.carbon.identity.oauth2.validators.DefaultOAuth2TokenValidator"	

OAuth.TokenValidators	jwt	[oauth.token_validator.jwt]
		enable = true
		class = "org.wso2.carbon.identity.oauth2.validators.OAuth2JWTTokenValidator"	

OAuth.TokenValidators	custom token validator	[[oauth.custom_token_validator]]
		type = "foo"
		class = "com.example.identity.TokenValidator"	

OAuth.ScopeValidators	jdbc	[oauth.scope_validator.jdbc]
		enable = true
		class = "org.wso2.carbon.identity.oauth2.validators.JDBCScopeValidator"	

OAuth.ScopeValidators	xacml	[oauth.scope_validator.xacml]
		enable = true
		class = "org.wso2.carbon.identity.oauth2.validators.xacml.XACMLScopeValidator"	

OAuth.ScopeValidators	custom scope validator	[oauth.custom_scope_validator] class = "org.wso2.carbon.identity.oauth2.validators.xacml.XACMLScopeValidator"	
OAuth.EnableAssertions	***UserName***	[oauth.token_generation] include_username_in_access_token= true	

OAuth	***BuildSubjectIdentifierFromSPConfig***	[oauth]
		validation_response_subject_identifier_format= "fqn"#"app_configured"
			if validation_response_subject_identifier_format= fqn then BuildSubjectIdentifierFromSPConfig= false
			if validation_response_subject_identifier_format= app_configured then BuildSubjectIdentifierFromSPConfig = true

OAuth.AuthorizationContextTokenGeneration	***Enabled***	[oauth.token.validation] include_validation_context_as_jwt_in_reponse=	
	TokenGeneratorImplClass	[oauth.extensions] token_context_generator=	
	ClaimsRetrieverImplClass	[oauth.extensions] token_context_claim_retriever=	
	ConsumerDialectURI	[oauth.extensions] token_context_dialect_uri=	
	SignatureAlgorithm	[oauth.token_validation] validation_response_signing_algorithm=	
	AuthorizationContextTTL	[oauth.token.validation] validation_response_jwt_validity=	
OAuth.SAML2Grant	***UserType***	[oauth.grant_type.saml_bearer] user_type= "FEDERATED"	
OAuth.OpenIDConnect	***ConvertOriginalClaimsFromAssertionsToOIDCDialect***	[oauth.oidc.claims] enable_oidc_dialect=	
	AddUnmappedUserAttributes	[oauth.oidc.claims] enable_unmapped_user_attributes=	
	IDTokenBuilder	[oauth.oidc.extensions] id_token_builder=	
	SignatureAlgorithm	[oauth.oidc.id_token] signature_algorithm=	
	IDTokenEncryptionAlgorithm	[oauth.oidc.id_token] supported_encryption_algorithms=[]	First item from the list supported_encryption_algorithms will be configured as IDTokenEncryptionAlgorithm

	IDTokenEncryptionMethod	[oauth.oidc.id_token] supported_encryption_methods=[]	First item from the list supported_encryption_methods will be configured as IDTokenEncryptionMethod
	SupportedIDTokenEncryptionAlgorithms	[oauth.oidc.id_token] supported_encryption_algorithms=[]	
	SupportedIDTokenEncryptionMethods	[oauth.oidc.id_token] supported_encryption_methods=[]	
	Audiences	[oauth.oidc.id_token] audiences=[]	
	IDTokenIssuerID	[oauth.oidc.id_token] issuer=	
	IDTokenCustomClaimsCallBackHandler	[oauth.oidc.extensions] claim_callback_handler=	
	IDTokenExpiration	[oauth.oidc.token_validation] id_token_validity=	
	UserInfoJWTSignatureAlgorithm	[oauth.oidc.user_info] jwt_signature_algorithm=	
	UserInfoEndpointClaimRetriever	[oauth.oidc.extensions] user_info_claim_retriever=	
	UserInfoEndpointRequestValidator	[oauth.oidc.extensions] user_info_request_validator=	
	UserInfoEndpointAccessTokenValidator	[oauth.oidc.extensions] user_info_access_token_validator=	

	UserInfoEndpointResponseBuilder	[oauth.oidc]
		user_info.response_type= "json" #"jwt"
			if user_info.response_type= json then UserInfoEndpointResponseBuilder= "org.wso2.carbon.identity.oauth.endpoint.user.impl.UserInfoJSONResponseBuilder"
			if user_info.response_type= jwt then UserInfoEndpointResponseBuilder = "org.wso2.carbon.identity.oauth.endpoint.user.impl.UserInfoJWTResponse"

	SkipUserConsent	[oauth]
		consent_prompt= true
			if consent_prompt= true then SkipUserConsent= false
			if consent_prompt= false then SkipUserConsent = true

	SignJWTWithSPKey	[authentication]
		sign_auth_response_with_tenant_of= "sp" #user
			if sign_auth_response_with_tenant_of= "sp" then SignJWTWithSPKey = true
			if sign_auth_response_with_tenant_of= "user" then SignJWTWithSPKey = false

	LogoutTokenExpiration	[oauth.oidc.token_validation] logout_token_validity=	

	RequestObjectBuilder	[oauth.oidc.request_object_builder.request_param_value_builder]
		enabled = true
		class = "org.wso2.carbon.identity.openidconnect.RequestParamRequestObjectBuilder"	

	RequestObjectBuilder custom	[[oauth.oidc.custom_request_object_builder]]
		type = "foo"
		class = "com.example.oauth.CustomRequestBuilder"	

	RequestObjectValidator	[oauth.oidc.extensions.request_object_validator] request_object_validator =	
OAuth.TokenPersistence	***RetryCount***	[oauth.token_generation] retry_count_on_persistance_failures =	
OAuth	***RenewTokenPerRequest***	[oauth.token_renewal] renew_access_token_per_request =	
SSOService	***EntityId***	[saml] entity_id =	
	IdentityProviderURL	[saml.endpoints idp_url=	
	DefaultLogoutEndpoint	[saml.endpoints logout=	
	NotificationEndpoint	[saml.endpoints notification=	
	ArtifactResolutionEndpoint	[saml.endpoints artifact_resolution=	
	SingleLogoutRetryCount	[saml.slo] retry_attempts=	
	SingleLogoutRetryInterval	[saml.slo] retry_interval=	
	TenantPartitioningEnabled	[saml.tenant_partitioning] enable=	
	AttributesClaimDialect	[saml] attribute_claim_dialect=	
	SAMLSSOAssertionBuilder	[saml.extensions] assertion_builder=	
	SAMLSSOEncrypter	[saml.extensions] encrypter=	
	SAMLSSOSigner	[saml.extensions] signer=	
	SAML2HTTPRedirectSignatureValidator	[saml.extensions] redirect_signature_validator=	
	SAMLResponseValidityPeriod	[saml.response] validity=	
	UseAuthenticatedUserDomainCrypto	[saml] enable_user_domain_crpto=	
	SAMLDefaultSigningAlgorithmURI	[saml] signing_alg=	
	SAMLDefaultDigestAlgorithmURI	[saml] digest_alg=	
	SAMLDefaultAssertionEncryptionAlgorithmURI	[saml] assertion_encryption_alg=	
	SAMLDefaultKeyEncryptionAlgorithmURI	[saml] key_encryption_alg=	
	SLOHostNameVerificationEnabled	[saml.slo] host_name_verification=	
	SAML2ArtifactValidityPeriodInMinutes	[saml.artifact] validity=	
	SAMLECPEndpoint	[saml.endpoints] ecp=	
Consent	***EnableSSOConsentManagement***	[authentication.consent] prompt=	
SecurityTokenService	***IdentityProviderURL***	[sts.endpoint] idp=	
PassiveSTS	***IdentityProviderURL***	[passive_sts.endpoints] idp=	
	RetryURL	[passive_sts.endpoints] retry=	
	TokenStoreClassName	[passive_sts] token_store_class=	
	SLOHostNameVerificationEnabled	[passive_sts.slo] host_name_verification=	
EntitlementSettings.ThirftBasedEntitlementConfig	***EnableThriftService***	[entitlement.thrift] enable=	

	ReceivePort	[entitlement.thrift] receiver_port=	
	ClientTimeout	[entitlement.thrift] client_timeout=	
	KeyStore.Location	[entitlement.thrift.key_store] id=	
	KeyStore.Password	[entitlement.thrift.key_store] password=	
	ThriftHostName	[entitlement.thrift] hostname=	
SCIM	***UserEPUrl***	[scim.endpoints] users_endpoint=	
	GroupEPUrl	[scim.endpoints] groups_endpoint=	
SCIM.SCIMAuthenticators	***BasicAuthHandler***	[scim.authentication_handler.basic.properties] priority = 5	

	OAuthHandler	[scim.authentication_handler.oauth.properties]
		priority = 10
		authorization_server = "local://services"	

	Custom handler	[[scim.authenticator]]
		class = "com.example.identity.scim.Authenticator"
		priority = 10
		[scim.custom_authenticator.properties]
		AuthorizationServer = "http://auth.example.com"	

	ComplexMultiValuedAttributeSupportEnabled	[scim] enable_complex_multivalued_attribute_support=	
SCIM2	***UserEPUrl***	[scim2.endpoints] users_endpoint=	
	GroupEPUrl	[scim2.endpoints] groups_endpoint=	
	ComplexMultiValuedAttributeSupportEnabled	[scim2] enable_complex_multivalued_attribute_support=	
Recovery.ReCaptcha	***Password.Enable***	[identity_mgt.password_reset_email] enable_recaptcha=	
	Username.Enable	[identity_mgt.username_recovery.email] enable_recaptcha=	
Recovery.Notification	***Password.Enable***	[identity_mgt.password_reset_email] enable_password_reset_email=	
	Username.Enable	[identity_mgt.username_recovery.email] enable_username_recovery=	

	InternallyManage	[identity_mgt]
		email_sender= "internal" #"external"
			if email_sender= "internal" then InternallyManage= true
			if email_sender= "external" then InternallyManage = false

Recovery.Question	***Password.Enable***	[identity_mgt.password_reset_challenge_questions] enable_password_reset_challenge_questions =	
	Password.NotifyStart	[identity_mgt.password_reset_challenge_questions] notify_on_initiation =	
	Password.Separator	[identity_mgt.password_reset_challenge_questions] question_separator =	
	Password.MinAnswers	[identity_mgt.password_reset_challenge_questions] min_required_answers =	
	Password.ReCaptcha.Enable	[identity_mgt.password_reset_challenge_questions] enable_recaptcha =	
	Password.ReCaptcha.MaxFailedAttempts	[identity_mgt.password_reset_challenge_questions] failures_before_recaptcha =	
Recovery	***ExpiryTime***	[identity_mgt.password_reset_email] reset_mail_validity =	
	NotifySuccess	[identity_mgt.password_reset_email] notify_on_reset =	
Recovery.AdminPasswordReset	***Offline***	[identity_mgt.password_reset_by_admin] enable_offline_otp_based_reset =	
	OTP	[identity_mgt.password_reset_by_admin] enable_emailed_otp_based_reset =	
	RecoveryLink	[identity_mgt.password_reset_by_admin] enable_emailed_link_based_reset =	
EmailVerification	***Enable***	[identity_mgt.user_onboarding] enable_email_verification =	
	ExpiryTime	[identity_mgt.user_onboarding] verification_email_validity =	
	LockOnCreation	[identity_mgt.user_onboarding] lock_on_creation}=	

	Notification.InternallyManage	[identity_mgt]
		email_sender= "internal" #"external"
			if email_sender= "internal" then InternallyManage= true
			if email_sender= "external" then InternallyManage = false

	AskPassword.ExpiryTime	[identity_mgt.user_onboarding] ask_password_email_validity =	
	AskPassword.PasswordGenerator	[identity_mgt.user_onboarding] password_generator =	
SelfRegistration	***Enable***	[identity_mgt.user_self_registration] allow_self_registration=	
	LockOnCreation	[identity_mgt.user_self_registration] lock_on_creation=	

	Notification.InternallyManage	[identity_mgt]
		email_sender= "internal" #"external"
			if email_sender= "internal" then InternallyManage= true
			if email_sender= "external" then InternallyManage = false

	ReCaptcha	[identity_mgt.user_self_registration] enable_recaptcha=	
	VerificationCode.ExpiryTime	[identity_mgt.user_self_registration] verification_email_validity=	
SPRoleManagement	***ReturnOnlyMappedLocalRoles***	[sp_role_management] return_only_mapped_local_roles=	
EnableAskPasswordAdminUI		[identity_mgt.user_onboarding] ask_password_from_user=	
EnableRecoveryEndpoint		[identity_mgt.endpoint] enable_recovery_endpoint=	
EnableSelfSignUpEndpoint		[identity_mgt.endpoint] enable_self_signup_endpoint=	

AuthenticationPolicy	CheckAccountExist	[authentication_policy] check_account_exist=	
JITProvisioning	UserNameProvisioningUI	[authentication.jit_provisioning] username_provisioning_url=	
	PasswordProvisioningUI	[authentication.jit_provisioning] password_provisioning_url=	
EventListeners	org.wso2.carbon.user.mgt.workflow.userstore.UserStoreActionListe	[event.default_listener.workflow] priority= enable =	
	org.wso2.carbon.identity.mgt.IdentityMgtEventListener	[event.default_listener.identity_mgt] priority= enable =	
	org.wso2.carbon.identity.scim.common.listener.SCIMUserOperation	[event.default_listener.scim] priority= enable =	
	org.wso2.carbon.identity.scim2.common.listener.SCIMUserOperatio	[event.default_listener.scim2] priority= enable =	
	org.wso2.carbon.identity.governance.listener.IdentityStoreEventLis	[event.default_listener.governance_identity_store] priority= enable = data_store =	
	org.wso2.carbon.identity.governance.listener.IdentityMgtEventListe	[event.default_listener.governance_identity_mgt] priority= enable = data_store =	
	org.wso2.carbon.identity.data.publisher.oauth.listener.OAuthToken	[event.default_listener.oauth_listener] priority= enable =	
	org.wso2.carbon.user.mgt.listeners.UserDeletionEventListener	[event.default_listener.user_deletion] priority= enable =	
	org.wso2.carbon.identity.application.authentication.framework.han	[event.default_listener.consent_mgt_handler] priority= enable =	
	org.wso2.carbon.identity.application.authentication.framework.han	[event.default_listener.jit_provisioning_handler] priority= enable =	
	org.wso2.carbon.identity.application.authentication.framework.han	[event.default_listener.post_auth_association_handler] priority= enable =	
	org.wso2.carbon.identity.application.authentication.framework.han	[event.default_listener.subject_identifier_handler] priority= enable =	
	org.wso2.carbon.user.mgt.listeners.UserMgtAuditLogger	[event.default_listener.user_mgt_audit_logger] priority= enable =	
	org.wso2.carbon.user.mgt.listeners.UserManagementAuditLogger	[event.default_listener.user_management_audit_logger] priority= enable =	
	org.wso2.carbon.user.mgt.listeners.UserMgtFailureAuditLogger	[event.default_listener.user_failure_audit_logger] priority= enable =	
	org.wso2.carbon.user.mgt.listeners.UserClaimsAuditLogger	[event.default_listener.user_claim_audit_logger] priority= enable =	

	custom event listener	[[event_listener]]
		id = "custom_audit_listener"
		type = "org.wso2.carbon.identity.core.handler.AbstractIdentityMessageHandler"
		name = "com.example.identity.AuditListener"
		order = 99
		[event_listener.properties]
		file_name = "login_audit.log"	

<UserDeleteEventRecorders>	UserDeleteEventRecorder	[event.default_recorder.user_delete_event] name= enable = write_to_separate_csv.path =	

	custom event recorder	[[event_recorder]]
		name = "com.example.identity.Recorder"
		order = 99
		[event_recorder.properties]
		file_name = "login_audit.log"	

CacheConfig	AppAuthFrameworkSessionContextCache	[cache.framework_session_context_cache] enable = timeout = capacity =	
	AuthenticationContextCache	[cache.authentication_context_cache] enable = timeout = capacity =	
	AuthenticationRequestCache	[cache.authentication_request_cache] enable = timeout = capacity =	
	AuthenticationResultCache	[cache.authentication_result_cache] enable = timeout = capacity =	
	AppInfoCache	[cache.app_info_cache] enable = timeout = capacity =	
	AuthorizationGrantCache	[cache.authorization_grant_cache] enable = timeout = capacity =	
	JWKSCache	[cache.jwks_cache] enable = timeout = capacity =	
	OAuthCache	[cache.oauth_cache] enable = timeout = capacity =	
	OAuthScopeCache	[cache.oauth_scope_cache] enable = timeout = capacity =	

	OAuthSessionDataCache	[cache.oauth_session_data_cache] enable = timeout = capacity =	
	SAMLSSOParticipantCache	[cache.saml_sso_participant_cache] enable = timeout = capacity =	
	SAMLSSOSessionIndexCache	[cache.saml_sso_session_index_cache] enable = timeout = capacity =	
	SAMLSSOSessionDataCache	[cache.saml_sso_session_data_cache] enable = timeout = capacity =	
	ServiceProviderCache	[cache.service_provider_cache] enable = timeout = capacity =	
	ProvisioningConnectorCache	[cache.provisioning_connector_cache] enable = timeout = capacity =	
	ProvisioningEntityCache	[cache.provisioning_entity_cache] enable = timeout = capacity =	
	ServiceProviderProvisioningConnectorCache	[cache.service_provider_provisioning_connector_cache] enable = timeout = capacity =	
	IdPCacheByAuthProperty	[cache.idp_cache_by_auth_property] enable = timeout = capacity =	
	IdPCacheByHRI	[cache.idp_cache_by_hri] enable = timeout = capacity =	
	IdPCacheByName	[cache.idp_cache_by_name] enable = timeout = capacity =	
EnableFederatedUserAssociation		[user.association] enable_for_federated_users =	
<TenantContextsToRewrite>	***WebApp***	[tenant_context.rewrite] webapps= []	
	Servlet	[tenant_context.rewrite] servlets= []	
ClockSkew		[server] clock_skew=	
JWTValidatorConfigs	***Enable***	[oauth.jwks_endpoint] enable=	
	JWKSEndpoint.HTTPConnectionTimeout	[oauth.jwks_endpoint] connection_timeout=	
	JWKSEndpoint.HTTPReadTimeout	[oauth.jwks_endpoint] read_timeout=	
	JWKSEndpoint.HTTPSizeLimit	[oauth.jwks_endpoint] size_limit_bytes=	
<AdaptiveAuth>	***EventPublisher.ReceiverURL***	[authentication.adaptive] event_publisher.url=	

	EventPublisher.BasicAuthentication	[authentication.adaptive.event_publisher]
		authentication_type="basic"
			if authentication_type= "basic" then
			Enable= true
			username= admin
			password = admin
			url = https://localhost:8280

	AsyncSequenceExecutorPoolSize	[authentication.adaptive] async_executer_pool_size=""	
	MaxTotalConnections	[authentication.adaptive.http_connections] max=""	
	MaxTotalConnectionsPerRoute	[authentication.adaptive.http_connections] max_per_route=""	
	HTTPConnectionTimeout	[authentication.adaptive.http_connections] connection_timeout=""	
	HTTPReadTimeout	[authentication.adaptive.http_connections] read_timeout=""	
	HTTPConnectionRequestTimeout	[authentication.adaptive.http_connections] request_timeout=""	
	RefreshInterval	[authentication.adaptive.long_wait] page_refresh_interval=""	
	PromptOnLongWait	[authentication.adaptive.long_wait] prompt=""	
	LongWaitTimeout	[authentication.adaptive.long_wait] timout=""	
IntermediateCertValidation	***IntermediateCerts.CertCN***	[intermediate_cert_validation] cert_cns=[]	
	ExemptContext.Context	[intermediate_cert_validation] exempt_contexts=[]	
FederatedIDPRoleClaimValueAttributeSeparator		[federated.idp] role_claim_value_attribute_separator=""	
X509	***X509RequestHeaderName***	[x509] request_header_name=""	
LoggableUserClaims	***LoggableUserClaim***	[audit.log] loggable_user_claim=[]	
ConfigurationStore	***MaximumQueryLength***	[configuration.store] lquery_length.max=""	

EnableAudiences			
GrantTypeName			

	Custom ResourceAccessControl	[[resource.access_control]]
		context = "test1"
		secured = true
		http_method = "all"
		permissions = ["p1","p2"]
			If a user need to add a custom ResourceAccessControl section