Introduction to Sterling Integrator & File Gateway 6.1

Understanding the Backbone of B2B Integration and File Transfer

- **Enterprise-Grade Integration:** IBM Sterling B2B Integrator (SI) and Sterling File Gateway (SFG) 6.1 are enterprise-class platforms for secure, automated B2B data exchange across partners and systems.
- **Property Files at the Core:** Every system parameter—from ports and database settings to clustering and security—is controlled via structured property files.
- **Configuration-Driven Design:** The property file architecture enables modularity, repeatable deployments, and consistent configuration management across environments.
- **Purpose of This Study:** Provides a detailed understanding of file types, hierarchy, management procedures, and tuning strategies for SI and SFG administrators.

Property File Architecture Overview

Foundation of Configuration Management in Sterling 6.1

Configuration Backbone
Property files serve as the foundation of IBM Sterling's configuration framework—defining runtime, installation, and extension parameters.

Types of Files
Includes *.properties, *.properties_ext, *.properties.in, *.properties_ext.in, customer_overrides.properties, and sandbox.cfg.

Purpose & Usage
Initialization (*.in) files set baseline values; runtime (*.properties) files drive operations; overrides ensure persistence across upgrades.

Centralized Customization
sandbox.cfg and customer_overrides.properties act as control points for instance-level and persistent customizations.

Types of Property Files and Their Roles

Understanding File Functions in Sterling Configuration Lifecycle

- **Runtime Files (*.properties):** Active configuration files loaded during runtime; control database connectivity, ports, adapters, and performance settings.
- **Extension Files (*.properties_ext):** Provide modular extensions or overrides to base properties—ideal for customizing application-specific parameters.
- **Initialization Templates (*.in):** Used during installation to generate operational property files; define initial configuration states.
- **Master Override (customer_overrides.properties):** Centralized file preserving custom configurations across patches, upgrades, and reinstallation events.
- **sandbox.cfg:** Holds environment-level parameters like database credentials, ports, and cluster identifiers merged during setup.

Property File Hierarchy & Lifecycle

Understanding the Order of Precedence and Configuration Flow

- **Installation Phase:** Initialization files (*.in) generate operational *.properties files using setupfiles scripts; sandbox.cfg parameters are merged at this stage.
- **Runtime Phase:** The application loads *.properties files as active configurations defining all operational behaviors and system connections.
- **Override Phase:** customer_overrides.properties supersedes any prior values, ensuring consistent customization persistence through upgrades.
- **Parameter Substitution:** sandbox.cfg values are injected into initialization templates using & and ; delimiters, creating a unified runtime configuration.
- **Hierarchy Summary:** Precedence: *.properties.in → *.properties → customer_overrides.properties → sandbox.cfg merges.

Critical Configuration File: sandbox.cfg

Centralized Parameter Store for Installation-Specific Values

- **Purpose:** Acts as a master parameter source merged into initialization (.in) files to create final runtime property files.
- **Syntax:** Uses & and ; delimiters for parameter substitution, e.g., oraclePool.user=&ORA_USER; becomes oraclePool.user=oracle after merge.
- **Key Parameters:** Defines DB vendor, user credentials, ports (JMS, RMI, JMX), clustering attributes, and administrator contact details.
- **Modification Procedure:** Edit sandbox.cfg → run setupfiles script → restart Sterling B2B Integrator to propagate changes.
- **Best Practice:** Most parameters are not used at runtime; re-running setupfiles is required to reflect configuration updates.

Master Override File: customer_overrides.properties

Ensuring Persistent Custom Configurations Across Upgrades

- **Purpose and Role:** Acts as the single authoritative file for persisting all custom property modifications through upgrades, patches, and reinstallations.
- **Override Syntax:** Format: PROPERTY_FILE_NAME_PREFIX.PROPERTY_NAME=PROPERTY_VALUE. Prefix derived from servers.properties mapping.
- **Practical Examples:** Database tuning: jdbcService.oraclePool.max=150. LDAP config: securityService.LDAP_ENABLED=true.
- **Exclusions:** Certain files (archivethread, tuning, ui, partial security) do not support overrides—must edit .in files directly.
- **Best Practices:** Document all overrides, version control the file, back up before edits, and validate in lower environments first.

Key Configuration Files in Sterling Integrator

Core Files That Define System Behavior and Performance

- **jdbc.properties:** Controls all database connectivity, pooling parameters, and JDBC driver settings; critical for performance tuning and reliability.
- **noapp.properties:** Defines workflow persistence, scheduling, and system resource limits; major lever for throughput optimization.
- **servers.properties:** Maps prefixes to specific property files; essential for constructing correct override entries in customer_overrides.properties.
- **ops.properties:** Contains operational parameters for adapters, logging, and retry mechanisms.
- **security.properties:** Manages authentication, password policies, and session controls; some entries cannot be overridden for security reasons.

Sterling File Gateway Specific Property Files

Configuration Layers Unique to SFG Operations

- **Shared Infrastructure:** SFG operates on top of Sterling B2B Integrator, sharing its property file framework and override hierarchy.
- **SFG-Specific Overrides:** Use customer_overrides.properties to configure routing channels, file structure processing, and authentication settings.
- **Routing Channel Configuration:** Defines producer/consumer paths, file patterns, and routing facts for dynamic file movement across mailboxes.
- **File Structure Properties:** Describes layered file formats such as ZIP, GZIP, PGP, and Text; ensures producer and consumer compatibility.
- **Temporary File Management:** Parameters like routingService.IGNORE_TEMP_FILES=true prevent routing of incomplete files during transfers.

Property File Management Procedures

Controlled Methods for Editing, Applying, and Validating Configuration Changes

- **Procedure 1: Using .in Files:** Recommended for initialization-based configuration changes. Edit *.properties.in → run setupfiles → restart system.
- **Procedure 2: Using customer_overrides.properties:** Preferred method for upgrade-safe modifications. Enables persistent overrides without editing runtime files.
- **Procedure 3: Direct File Editing:** Use only when no .in or override option exists. Requires careful backup and documentation.
- **Procedure 4: Modifying sandbox.cfg:** Adjust environment parameters (ports, DB info) and re-run setupfiles to regenerate runtime properties.
- **Whitespace Handling:** Trim whitespace rigorously—leading or trailing spaces can cause misconfigurations or startup failures.

Common Configuration Scenarios & Use Cases

Practical Examples for System Administration and Optimization

- **Database Connection Pool Tuning:** Increase oraclePool.max/min and timeout parameters in customer_overrides.properties to expand concurrent connections.
- **Enabling LDAP Authentication:** Configure LDAP parameters in securityService and authentication_policy files for centralized user management.
- **Changing HTTP/HTTPS Ports:** Adjust PORT1 or B2B_HTTP_PORT in sandbox.cfg or override file to resolve port conflicts.
- **Performance Optimization:** Set persistence_level=PERSISTENCE_NONE, use FAST scheduling, and increase workflow threads for high throughput.
- **Cluster Configuration:** Define IS_CLUSTER=YES, PARTITION_NAME, and NODE_ID in sandbox.cfg; configure clusterControlService overrides per node.

Troubleshooting Property File Issues

Diagnosing and Resolving Common Configuration Failures

- **Property Changes Not Taking Effect:** Verify setupfiles execution, property prefix correctness, and restart sequence; ensure property supports override.
- **Database Connection Failures:** Check jdbc.properties syntax, credentials, driver availability, and database connectivity; validate via client test.
- **Performance Degradation:** Assess persistence_level, workflow threads, and connection pool size; rebalance based on system capacity.
- **Port Conflicts:** Identify occupied ports using netstat and adjust PORT1 or HTTP adapter configurations accordingly.
- **Cluster Sync Issues:** Ensure consistent PARTITION_NAME, unique NODE_ID, and valid JMS/JNDI ports across nodes.

Advanced Topics: Encryption, Custom Files, and Validation

Enhancing Security, Extensibility, and Governance

- **Custom Property Files:** Administrators can define new property files for application-specific settings, mapped in servers.properties for runtime access.
- **Property Encryption:** Sensitive fields (passwords, keys) can be encrypted using IBM utilities like encrypt_string.sh; decrypted automatically at runtime.
- **Validation Scripts:** Custom shell scripts can verify property file integrity, missing files, and syntax issues before deployment.
- **Backup Strategy:** Automate daily backups of properties directory with version control integration (Git/SVN). Retain history for rollback.
- **Upgrade Management:** Before upgrades, back up properties, test overrides, and merge new .in parameters post-upgrade to maintain consistency.

Best Practices & Production Checklist

Ensuring Reliability, Security, and Change Control in Sterling Environments

- **Use Overrides Strategically:** Always apply changes via customer_overrides.properties to preserve configuration persistence across upgrades.
- **Follow Change Management:** Test in lower environments, schedule maintenance windows, and document change approvals and rollback plans.
- **Backup & Version Control:** Maintain backups of properties and sandbox.cfg; integrate with Git/SVN for historical tracking.
- **Whitespace & Syntax Vigilance:** Avoid trailing spaces and misaligned property syntax; validate after every edit.
- **Operational Checklist:** Pre-change validation, backup confirmation, monitoring setup, and post-change verification must all be completed.

Summary & Conclusion

Mastering Property File Management in Sterling Integrator and File Gateway

- **Central Role of Property Files:** Property files define every aspect of Sterling configuration—from installation parameters to runtime tuning.
- **Hierarchy and Control:** Understanding file precedence ensures stable configuration management and prevents conflicts during deployment.
- **customer_overrides.properties Advantage:** Centralized, upgrade-safe file that sustains all environment-specific modifications.
- **Procedural Discipline:** Running setupfiles, validating syntax, and maintaining backups are key to avoiding runtime issues.
- **Operational Excellence:** With documentation, version control, and automation, administrators can achieve reliable, secure B2B operations.